Android HTML Tag Support

Android supports a limited set of HTML tags both natively and with the Html class (*Html.java*). The following explains this support.

HTML Class Tag Support

The Android *Html* class supports the following HTML tags and properties.

Note on the AndroidX HtmlCompat class: HtmlCompat calls through to the *Html* class, so *HtmlCompat* supports all the same tags as *Html* but ignores the *flags* argument for API versions below 24. You can use HtmlCompat for all versions of Android without having to check API level. Just keep in mind that the flags argument will be ignored for API levels below 24 and that certain tags are supported only on API 24 and above as noted in this document.

The following information is up to date as of API 34.

Android HTML Tag Support

Tags Supported by the Html Class

Tag	Block-level?[1]	Limited Style?[2]	Supported by Framework [3]	Span	Notes
<a>			**Y**	URLSpan(href)	Supports the *href* tag.
<b>			Y	StyleSpan(Typeface.BOLD, fontWeightAdjustment[5])	StyleSpan(Typeface.BOLD) before API 33; Native support for fontWeightAdjustment began with API 31
<big>			Y	RelativeSizeSpan(1.25f)	
<blockquote>	Y			QuoteSpan()	
 				Inserts '\n'	
<cite>				StyleSpan(Typeface.ITALIC)	
<del>				StrikethroughSpan()	Supported API 24+.
<dfn>				StyleSpan(Typeface.ITALIC)	
	Y				
				StyleSpan(Typeface.ITALIC)	
<font>			**Y**	*TypefaceSpan(face)*	*Supports the color and face attribute[4].*
<h1> … <h6>	Y			RelativeSizeSpan(size), StyleSpan(Typeface.BOLD, fontWeightAdjustment[5])	Size is 1.5f, 1.4f, 1.3f, 1.2f, 1.1f, or 1f for headings h1-h6 RelativeSizeSpan(size), StyleSpan(Typeface.BOLD) before API 33
<i>			Y	StyleSpan(Typeface.ITALIC)	
<img>				ImageSpan(d, src)	d-drawable; src-source. *src* tag with *Html.ImageGetter*
	Y	Y	Y	BulletSpan()	Supported API 24+.
<p>	Y	Y			
<s>				StrikethroughSpan()	Supported API 24+.
<small>			Y	RelativeSizeSpan(0.8f)	
		Y			Supported API 24+.
				StyleSpan(Typeface.BOLD, fontWeightAdjustment[5])	StyleSpan(Typeface.BOLD) before API 33
<strike>			Y	StrikethroughSpan()	Supported API 24+.
<sub>			Y	SubscriptSpan()	
<sup>			Y	SuperscriptSpan()	

Android HTML Tag Support

Tag	Block-level?[1]	Limited Style?[2]	Supported by Framework [3]	Span	Notes
<tt>			Y	TypefaceSpan("monospace")	
<u>			Y	UnderlineSpan()	
	Y				Supported API 24+.
Other tags					Supported *with Html.TagHandler*.

[1] Block-level elements support the text-align style property. The supported values for text-align are: “start”, ”center” and ”end”. (”justify” is not supported.) Text spans are Alignment(Layout.Alignment.ALIGN_NORMAL) (”start”), Alignment(Layout.Alignment.ALIGN_CENTER) (”center”) and Alignment(Layout.Alignment.ALIGN_OPPOSITE) (”end”).

[2] “Limited style” indicates that the tag supports the “color”, “background[-color]” and “text-decoration” properties. The only supported value for” text-decoration” is “line-through”. See below for details on color support. Text spans used for styles are ForeGround (color), Background (background[-color]) and StrikeThrough (text-decoration=”line-through”).

[3] Natively supported by the Android framework (string resources).

[4] face can be any typeface name supported by the *TypefaceSpan* class.

[5] fontWeightAdjustment was added in API 31 but not used in the *Html* class until API 33.

Html.fromHtml() Flags

Values for the *flags* argument of *Html.fromHtml()* are:

FROM_HTML_SEPARATOR_LINE_BREAK_BLOCKQUOTE
FROM_HTML_SEPARATOR_LINE_BREAK_DIV
FROM_HTML_SEPARATOR_LINE_BREAK_HEADING
FROM_HTML_SEPARATOR_LINE_BREAK_LIST
FROM_HTML_SEPARATOR_LINE_BREAK_LIST_ITEM
FROM_HTML_SEPARATOR_LINE_BREAK_PARAGRAPH

Each of the preceding flags specifies that the HTML processor should add a single newline after each named block-level element. If the flag is not set, the processor adds two newlines which is the legacy behavior.

For example: Setting FROM_HTML_SEPARATOR_LINE_BREAK_HEADING will add one newline after a heading (<h1>, <h2>, etc.)

FROM_HTML_MODE_LEGACY: If this flag is set, then two newlines will be added after each block-level element. Setting this flag is the same as passing zero.

FROM_HTML_MODE_COMPACT: Use of this flag is the same as specifying all of the line break flags which will remove all extra newlines from block-level elements.

FROM_HTML_OPTION_USE_CSS_COLORS: For named colors, use the CSS numeric values instead of the values defined by the Android *Color* class.

For instance, if "darkgray" is specified as the color and this flag is set then the color value will be the CSS value for “darkgray” (0xFFA9A9A9) instead of the value for “darkgray” defined in the *Color* class (0xFF444444). If this flag is not set, then the value will be the value from the *Color* class.

Android HTML Tag Support

Android Native Support for HTML Tags

"stringblock.java" in the Android Open Source Project (AOSP) contains the processing logic for Android string resources and is the reference for this section.

For simple text formatting, Android natively supports a subset of HTML tags. Used in a string resource, these are interpreted by the framework. To retrieve a styled string from a string resource that contains any of these tags, call *CharSequence getText (int resId)* In the *Context* class.

HTML tags that are specified in a string resource that are not named below will not cause an error but will be ignored and stripped. (See *StringBlock.java* for details.)

Tags without Attributes

Simple Tag	Text Span Used
<annotation>	Annotation spans of key/value pairs. See documentation for Annotation spans.
<b>	StyleSpan(Typeface.BOLD, fontWeightAdjustment) is not supported.) Before API 31: StyleSpan(Typeface.BOLD)
<big>	RelativeSizeSpan (1.25f) 25% larger text
<i>	StyleSpan(Typeface.ITALIC)
	BulletSpan(10) (is not supported. See note below.)
<marquee>	Span is set to "TextUtils.TruncateAt.MARQUEE" (See note below.)
<small>	RelativeSizeSpan(0.8f) 20% smaller text
<strike>	StrikethroughSpan()
<sub>	SubscriptSpan()
<sup>	SuperscriptSpan()
<tt>	TypefaceSpan("monospace")
<u>	UnderlineSpan()

Tags with Attributes

- Font: <font> *("int" is an integer value, For "color", see the notes.)*
    - bgcolor="color" Applies a *BackgroundColorSpan* to the text.
    - color="color" Applies a *ForegroundColorSpan* to the text.
    - face="font_family" "font_family" can be any font acceptable to *TypefaceSpan*.

  - fgcolor="color" Applies a *ForegroundColorSpan* to the text.
  - height="int" Applies an internally defined "*Height*" span that forces the text to be a specific height in pixels.
  - size="int" Applies an *AbsoluteSizeSpan* to the text.
- Link: <a> (*URLSpan(href)*)
  - href="url"
- Annotation: <annotation> (*Annotation(key, value)*)
  - key="value" Any number of key/value pairs where key can be any reasonable string. Each key/value pair creates a new Annotation span. Because of the way that Android encodes styles, "value" cannot contain a semicolon (";").

All spans are set with the flag "Spannable.SPAN_EXCLUSIVE_EXCLUSIVE" except for "marquee" which is set with the flag "Spannable.SPAN_INCLUSIVE_INCLUSIVE".

Some Tags that are not Supported Natively

The following tags are mentioned as supported in the documentation but do not seem to be natively supported by Android.

- Line breaks:
 *(Use "\n" instead.
 is accepted by Android Studio but has no effect.
 may be processed correctly by Android, but the string resource editor of Android Studio will not accept it as of Android Studio Hedgehog | 2023.1.1 Patch 1)*
- CSS style:
- Paragraphs: <p dir="rtl | ltr" style="…">
- Division:

Notes on Native Android Support of HTML Tags

The unordered list tag, <ul>, is ignored and the <li> tag can stand on its own. The <li> tag does not introduce a line break as might be expected and should be preceded by a newline. The radius of the bullet created is 4px, the gap width is 10px and the bullet color is black (0xFF000000). These values cannot be changed.

For <marquee>, Android creates a "TextUtils.TruncateAt.MARQUE" span for the enclosed text. Other than to tag the text as "marquee" for some use later, it is unclear if Android does anything else with this tag. Note that Html.fromHtml() does not support the <marquee> tag.

For details on what is documented as supported, see
https://developer.android.com/guide/topics/resources/string-resource#StylingWithHTML

Android HTML Tag Support

For supported tags, reference is made to the following methods in StringBlock.java from the AOSP:

applyStyles()
https://cs.android.com/android/platform/superproject/main/+/main:frameworks/base/core/java/android/content/res/StringBlock.java;l=230?q=stringblock.java&ss=android%2Fplatform%2Fsuperproject%2Fmain

getColor()
https://cs.android.com/android/platform/superproject/main/+/main:frameworks/base/core/java/android/content/res/StringBlock.java;drc=38c85ea37800d1d8a1d850bafd9229f46f5704a9;l=392?q=stringblock.java&ss=android%2Fplatform%2Fsuperproject%2Fmain

String resources to test native support:

```
<string name="html_test">These tags will format:\n
<b>bold</b>\n<i>italic</i>\n<big>big</big>\n<small>small</small>\n<tt>
monospace</tt>
\n<strike>strike (strike)</strike>\n<u>underline</u>\nA<sup>superscript</sup>
\nA<sub>subscript</sub>\n<li>Bullet 1</li>\n<li>Bullet 2</li>\n\n
<font bgcolor="#ff000000" color="@red" face="serif" height="20" size="30">Font
height</font>
\n<a href="https://google.com">Link</a>\n
\nThese tags will not format:
<br/><br/><br/>
\n<em>emphasis</em>\n<cite>cite</cite>\n<dfn>dfn</dfn>\n<del>del</del>\n<s>strike
(s)</s>
\n<marquee>marquee</marquee>\n<p style="fgcolor:#ff0000">Paragraph</p>
\n<span style="color:#FF0000 background_color: #00FF00">span</span> <div>div</div>
</string>
```

And the associated Kotlin code in an Activity:

```
var s = getText(R.string.html_test)
binding.textView.setText(s)
binding.textView.movementMethod = LinkMovementMethod.getInstance()

// To inspect the spans created
var spans = (s as Spanned).getSpans(0, s.length, Any::class.java)
```

String Resources: Html.fromHtml() vs. Native Android HTML Support

Android strips all tags from a string resource and stores them separately from the string. For those tags that Android natively supports, `Context.getText(resource_id)` will retrieve the string and apply the formatting that the supported tags call for by creating the required spans. Those tags that Android does not support natively are simply lost. For instance, for the string resource

```
<string name="test1"><s>Strike</s> <b>Bold</b></string>
```

`textView.setText(getText(R.string.test1))` will display

Strike **Bold**

because the bold tag `<b>` is supported but the strike through tag `<s>` is not and is stripped.

It is important to note that `Context.getString(resource_id)` *will return a string stripped of all tags with no formatting.*

`Html.fromHtml(string, flags)` handles more tags than native Android. Since the first argument is a string, it will not have any formatting and all tags will be stripped if the source is a string resource. So, to use tags that Android does not natively support without being stripped, the string resource must be modified to allow the pass-through of tags. This is easily done by simply replacing the less than sign of the tag with the HTML entity "`&lt;`". This, in essence, "hides" the tag from Android resource packaging. Thus, the "test1" string above becomes

```
<string name="test2">&amp;lt;s&gt;Strike&amp;lt;/s&gt; &amp;lt;b&gt;Bold&amp;lt;/b&gt;</string>
```

`getString(R.string.test2)` will return "`<s>Strike</s> <b>Bold</b>`" so, `textView.setText(Html.fromHtml(getString(R.string.test2), 0))`

will display as ~~Strike~~ **Bold**

`getText` will work as `getString` does above with the "<" but will also interpret other tags that are natively supported but not escaped. Since `Html.fromHtml` supports all natively supported tags (except marquee) as well as additional tags, it is unclear why one would want to mix native support and Html class support.

Replacing less than signs with HTML entities can be tedious. A way to escape the entire resource string so that all tags are passed using `Context.getString()` is to enclosed the string in a `CDATA` section as follows:

```
<string name="test3"><![CDATA[<s>Strike</s> <b>Bold</b>]]></string>
```

`Context.etString(R.string.test3)` will return

```
<s>Strike</s> <b>Bold</b>
```

So, a call to `textView.setText(Html.fromHtml(getString(R.string.test3),0))` will display

~~Strike~~ **Bold**

which is what we want.

HTML Color Support

Color Class Colors

Colors defined in the Color class that are used by the Android framework and the Html class:

```
aqua:       0xFF00FFFF
black:      BLACK
blue:       BLUE
cyan:       CYAN
darkgray:   DKGRAY
darkgrey:   DKGRAY
fuchsia:    0xFFFF00FF
gray:       GRAY
green:      GREEN
grey:       GRAY
lightgray:  LTGRAY
lightgrey:  LTGRAY
lime:       0xFF00FF00
magenta:    MAGENTA
maroon:     0xFF800000
navy:       0xFF000080
olive:      0xFF808000
purple:     0xFF800080
red:        RED
silver:     0xFFC0C0C0
teal:       0xFF008080
white:      WHITE
yellow:     YELLOW

(Constants used above)
BLACK:           0xFF000000
BLUE:            0xFF0000FF
CYAN:            0xFF00FFFF
DKGRAY:          0xFF444444
GRAY:            0xFF888888
GREEN:           0xFF00FF00
LTGRAY:          0xFFCCCCCC
MAGENTA:         0xFFFF00FF
RED:             0xFFFF0000
TRANSPARENT:     0
WHITE:           0xFFFFFFFF
YELLOW:          0xFFFFFF00
```

The following colors definitions are used by the Html class if the Html.FROM_HTML_OPTION_USE_CSS_COLORS flag is set (API 24+):

```
darkgray:       0xFFA9A9A9
darkgrey:       0xFFA9A9A9
gray:           0xFF808080
green:          0xFF008000
grey:           0xFF808080
lightgray:      0xFFD3D3D3
lightgrey:      0xFFD3D3D3
```

Although "white" is defined as a valid color in the Color class, its value (0xFFFFFFFF) causes processing to ignore the color altogether. This is because the value returned for "white" by the Color class (0xFFFFFFFF) is interpreted as a "not found" condition (-1).

A work-around is to specify 0xFFFFFF for the color "white" and let the processing add the leading "FF".

How to Specify Colors with the Html Class

HTML colors can take one of the following formats for strings passed to Html.fromHtml(). Note that how color is handled differs between the Html class and native Android.

1. #AARRGGBB (AA = alpha; RR = red; GG = green; BB = blue),
2. #RRGGBB,
3. 0xAARRGGBB
4. 0xRRGGBB
5. a decimal number preceded by an optional minus sign, plus sign or a
6. color name from the *Colors* class.

The resultant color will always have the high-order byte forced to *0xFF*. Due to an internal error in the *Color* class, it is probably best to avoid the #AARRGGBB format since no color will register if the high-order bit is set.

Some examples follow. Strings are retrieved using *Context.getText(stringResId)*.

Android HTML Tag Support

```
<string name="color_test_html_1"><![CDATA[<font color="#FF0000">I am red. ("#FF0000")</font>]]></string>
<string name="color_test_html_2"><![CDATA[<font color="#7F0000ff">I should be translucent blue but am not. ("#7F0000ff")</font>]]></string>
<string name="color_test_html_3"><![CDATA[<font color="#8FFF0000">I should be translucent red but am not translucent or red. ("#8FFF0000")</font>]]></string>
<string name="color_test_html_4"><![CDATA[<font color="0x00ff00">I am green. ("0x00ff00")</font>]]></string>
<string name="color_test_html_5"><![CDATA[<font color="teal">I am teal. ("teal")</font>]]></string>
<string name="color_test_html_6"><![CDATA[<font color="16711680">I am red. ("16711680")</font>]]></string>
<string name="color_test_html_7"><![CDATA[<font color="-65536">I am red. ("-65536")</font>]]></string>
```

Android HTML Tag Support

How to Specify Colors for the Android Framework

An Android resource string HTML color can take one of the following formats:

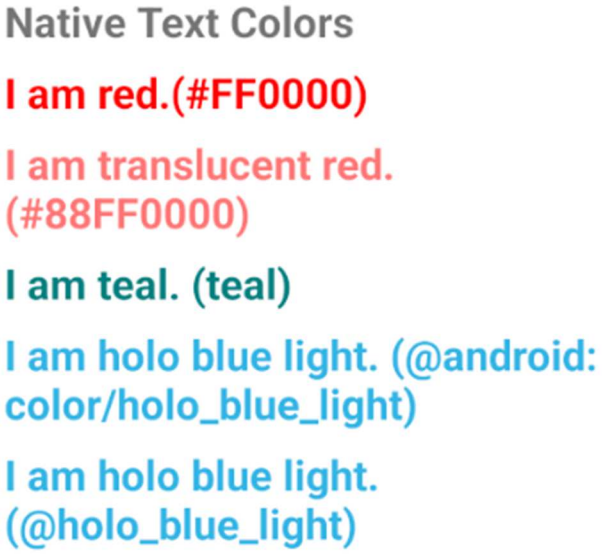
1. #AARRGGBB or #RRGGBB (AA = alpha; RR = red; GG = green; BB = blue) For CSS, the alpha value appears last (#RRGGBBAA). But, for Android, the alpha value appears first.

2. A color name from Colors.java (See **Color Class Colors)**, or,

3. "@" followed by an Android resource color name

Some examples:

```
<string name="color_test_native_1"><font color="#FF0000">I am red.("#FF0000")</font></string>
<string name="color_test_native_2"><font color="#88FF0000">I am translucent red. ("#88FF0000")</font></string>
<string name="color_test_native_3"><font color="teal">I am teal. ("teal")</font></string>
<string name="color_test_native_4"><font color="@android:color/holo_blue_light">I am holo blue light. (@android:color/holo_blue_light)</font></string>
<string name="color_test_native_5"><font color="@holo_blue_light">I am holo blue light. (@holo_blue_light)</font></string>
```



###