

FreePascal 与 Lazarus

颜色

目录

| | |
|--|---|
| 概述 | 2 |
| 在 TColor 和 RGB 值之间转换..... | 2 |
| 转换 TColor 到/来自 string(字符串)..... | 2 |
| 转换 TColor 到/来自 HTML string(字符串) #rrggbb..... | 2 |
| 标准颜色表..... | 2 |
| 系统颜色 | 3 |
| 示例 : clInfoBk, clInfoText | 3 |
| Theme(主题) 更改 | 4 |
| 系统颜色表 | 4 |
| 在你 的自定义控件上绘制 theme(主题)元素..... | 6 |

翻译 : robsean

翻译时间 : 2018 年 12 月 19 日---2018 年 12 月 19 日

www.Lazaruscn.top

概述

在 LCL 中，TColor 是标准颜色类型。它和 Delphi 的 TColor 兼容。TColor 可以表示要么 一个 RGB (3x8 比特)值，要么一个 **system** 颜色，像 clDefault。LCL 也可以和 fpImage 系统一起工作，使用 TFPColor 类型(它是 RGBA(4x16 比特)，非 RGB(3x8 比特)，像 TColor)。

在 TColor 和 RGB 值之间转换

Graphics 单元提供提供下面的函数：

```
function Blue(rgb: TColor): BYTE; // does not work on system color
function Green(rgb: TColor): BYTE; // does not work on system color
function Red(rgb: TColor): BYTE; // does not work on system color
function RGBToColor(R, G, B: Byte): TColor;
procedure RedGreenBlue(rgb: TColor; out Red, Green, Blue: Byte); // does not work on system color
function FPColorToTColor(const FPColor: TFPColor): TColor;
function TColorToFPColor(const c: TColor): TFPColor; // does not work on system color
function IsSysColor(AColor: TColorRef): Boolean;
```

转换 TColor 到/来自 string(字符串)

函数来转换字符串像"25500"或"\$AA0088"或"clNavy"到 TColor：

- StringToColor
- StringToColorDef

转换 TColor 到一个友好的字符串像"clNavy"或"\$AA0002"：

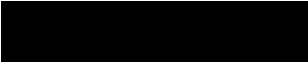


- ColorToString

转换 TColor 到/来自 HTML string(字符串) #rrggbb

查看代码在[转换颜色到/来自 HTML](#)。

标准颜色表

大约 20 种预定义颜色常量被提供，它们是与 Delphi-兼容的：

| 颜色常量 | 意思 | TColor 使用十六进制值 | 示例 |
|----------|-----|-------------------|--|
| clBlack | 黑色 | TColor(\$000000); |  |
| clMaroon | 褐红色 | TColor(\$000080); |  |
| clGreen | 绿色 | TColor(\$008000); |  |

| | | | |
|--------------|-----|-------------------|--|
| clOlive | 橄榄绿 | TColor(\$008080); |  |
| clNavy | 深蓝色 | TColor(\$800000); |  |
| clPurple | 紫色 | TColor(\$800080); |  |
| clTeal | 蓝绿色 | TColor(\$808000); |  |
| clGray | 灰色 | TColor(\$808080); |  |
| clSilver | 银色 | TColor(\$C0C0C0); |  |
| clRed | 红色 | TColor(\$0000FF); |  |
| clLime | 酸橙绿 | TColor(\$00FF00); |  |
| clYellow | 黄色 | TColor(\$00FFFF); |  |
| clBlue | 蓝色 | TColor(\$FF0000); |  |
| clFuchsia | 紫红色 | TColor(\$FF00FF); |  |
| clAqua | 湖绿色 | TColor(\$FFFF00); |  |
| clLtGray | 浅灰色 | TColor(\$C0C0C0); | clSilver 别名 |
| clDkGray | 深灰色 | TColor(\$808080); | clGray 别名 |
| clWhite | 白色 | TColor(\$FFFFFF); | |
| clCream | 奶油色 | TColor(\$F0FBFF); | Lazarus 1.2 和较新的 |
| clMedGray | 中灰色 | TColor(\$A4A0A0); | Lazarus 1.2 和较新的 |
| clMoneyGreen | 薄荷绿 | TColor(\$C0DCC0); | Lazarus 1.2 和较新的 |
| clSkyBlue | 天蓝色 | TColor(\$F0CAA6); | Lazarus 1.2 和较新的 |

系统颜色

示例：clInfoBk, clInfoText

系统颜色是带有一个指定意义的颜色常量。它们真实的值依赖于背景(context)和 theme(主题)。它们不简单的颜色。例如 clInfoBk:

```
Form1.Canvas.Brush.Color:=clInfoBk; // use the default background brush of a hint window
Form1.Canvas.FillRect(10,10,50,50);
```

在微软 Windows 系统上，一个提示可能有一种白色背景，所以上面将绘制白色。在 Linux/gtk2 系统上，它可能是一种金属纹理，使用上面将绘制纹理。如果你想在这上面放置一些文本，你需要相对应的颜色，像 clInfoText，否则，你的文本可能对用户不可读。例如：

```
Form1.Canvas.Brush.Color:=clInfoBk; // use the default background brush of a hint window
Form1.Canvas.FillRect(10,10,50,50);
```

```
Form1.Canvas.Font.Color:=clInfoText; // use the default text color of a hint window
Form1.Canvas.TextOut(10,10,'Hint');
```

系统颜色 **clInfoBk** 不能被使用于 **Pen.Color** 和 **Font.Color**。如果你这么做，结果 的未定义的，并依赖于 widgetset 和用户 theme（主题）。对于 **clInfoText** 相同：它仅 能被作为 **Font.Color** 使用。作为 **Brush.Color** 使用它，可能不工作。此刻，所有 widgetsets 也 允许作为 **Pen.Color** 使用它。

Theme(主题) 更改

当用户切换主题时系统颜色更改。一个 **clInfoBk** 可能更改，从白色到蓝色或从一种颜色到一种纹理(texture)。当你分配一个新的 Brush 句柄时，这种更改将发生。 记住，一个简单的指派 **Brush.Color:=clInfoBk** 不分配一个 Brush 句柄。Brush 句柄在使用上是被分配的。例如：

```
Form1.Canvas.Brush.Color:=clInfoBk; // this will not create a new brush handle
Form1.Canvas.FillRect(10,10,50,50); // this will create the brush handle with the currently active theme brush for hint windows
...
// if the theme changes in this moment the Brush.Handle is still allocated with the old values
...
Form1.Canvas.FillRect(10,10,50,50); // this will paint with the old theme brush
Form1.Canvas.Brush.Color:=clInfoBk; // assigning the old value will not create a new brush handle
Form1.Canvas.FillRect(10,10,50,50); // this will paint with the old theme brush
Form1.Canvas.Brush.Color:=clRed;    // assigning a new color, old Handle invalid
Form1.Canvas.Brush.Color:=clInfoBk; // assigning a new color, old Handle invalid
Form1.Canvas.FillRect(10,10,50,50); // this will create a new handle and paint with the new theme
```

系统颜色表

下面表格列出系统颜色和它们的意思。在定义之外使用它们，结果依赖于 widgetset 和 theme（主题）。例如 **clDefault** 是使用设备描述表的一般的背景。如果你想要在你自己拥有的自定义控件上使用单元 **Themes** 的绘画函数来 绘涂按钮元素。

| 常量 | LCL 定义 | Delphi 注释 | 支持的 Widgetsets |
|-----------|---|-----------|----------------|
| clNone | 标示 "不做涂染"。使用它作为控件的颜色是未定义的。控件将不会显而易见地获得。 | - | 全部 |
| clDefault | 对 Brush 使用它将使用目标 DC(设备描述表)的一般的背景。 <ul style="list-style-type: none"> 在一个窗体上用画布覆盖一个 <i>FillRect</i> ,将绘涂一个带有标准窗体的 | - | 全部 |

| | | | |
|-------------------|---|---|----|
| | <p>一般背景的矩形区域。这是诸如此类的 widgetset 和 theme (主题) 定义。这可能是灰色或一个渐变色或一个图片。</p> <ul style="list-style-type: none"> • 在一个 TListBox 的画布上使用 clDefault 将带有一般比较来绘涂，在 Windows 上，这是一般的白色。所以在一个 TListBox clDefault 中是和 clWindow 系统的。 • 使用它作为 Pen 颜色将使用设备描述表的默认行颜色。 • 使用它作为 Font 颜色将使用设备描述表的一般文本颜色。 | | |
| clScrollBar | 滚动条体 | - | 全部 |
| clBackground | 桌面背景颜色 | - | 全部 |
| clActiveCaption | 激活窗体标题栏 | - | 没有 |
| clInactiveCaption | 非激活窗体标题栏 | - | 没有 |
| clMenu | 常规菜单项目背景颜色 r | - | 没有 |
| clWindow | 未选择文本的一般背景 brush。被定义用于控件，像 TEdit，TComboBox，TMemo，TListBox，TTreeView. | - | 没有 |
| clWindowFrame | 窗体边框的颜色 | - | 没有 |
| clMenuText | 字体颜色和 clMenu 一起来使用 | - | 没有 |
| clWindowText | 字体颜色和 clWindow 一起来使用 | - | 没有 |
| clCaptionText | 激活窗体标题栏文本颜色 | - | 没有 |
| clActiveBorder | ? | - | 没有 |
| clInactiveBorder | ? | - | 没有 |
| clAppWorkspace | MDIMain 窗体背景颜色 | - | 没有 |
| clHighlight | 选择元素的 brush 颜色 | - | 没有 |
| clHighlightText | 选择的文本字体颜色(和 clHighligh 一起使用). | - | 没有 |
| clBtnFace | 按钮背景颜色 | - | 没有 |

| | | | |
|---------------------------|----------------------------------|---|----|
| clBtnShadow | 按钮阴影颜色（右下角）用于实现 3D 效果 | - | 没有 |
| clGrayText | 不使用元素的字体颜色 | - | 没有 |
| clBtnText | 按钮字体颜色和 clBtnFace 一起来使用 | - | 没有 |
| clInactiveCaptionText | 非激活窗体标题栏文本颜色 | - | 没有 |
| clBtnHighlight | 按钮高亮颜色（左上角）用于实现 3D 效果 | - | 没有 |
| cl3DDkShadow | ? | - | 没有 |
| cl3DLight | ? | - | 没有 |
| clInfoText | 用于提示的字体颜色。和 clInfoB 一起使用 | - | 全部 |
| clInfoBk | 用于提示的 Brush 颜色。和 clInfoText 一起使用 | - | 全部 |
| clHotLight | ? | - | 没有 |
| clGradientActiveCaption | 激活窗体标题栏的第二种被用于制作渐变色的颜色 | - | 没有 |
| clGradientInactiveCaption | 非激活窗体标题栏的第二种被用于制作渐变色的颜色 | - | 没有 |
| clMenuHighlight | 选择的菜单项目的背景颜色 | - | 没有 |
| clMenuBar | 菜单栏的背景颜色 | - | 没有 |
| clForm | ? | - | 没有 |
| clColorDesktop | ? | - | 没有 |
| cl3DFace | ? | - | 没有 |
| cl3DShadow | ? | - | 没有 |
| cl3DHiLight | ? | - | 没有 |
| clBtnHiLight | 和 clBtnHighlight 相同 | - | 没有 |

在你的自定义控件上绘制 **theme(主题)**元素

使用单元 **Themes** 提供函数来绘制标准控件的单个元素。例如，使用下列代码来绘制一个扩展符号，像一个 TTreeView:

```
uses Themes;

...

procedure TYourCustomControl.Paint;
```

```

const
  PlusMinusDetail: array[Boolean {Hot}, Boolean {Expanded}] of TThemedTreeview =
  (
    (ttGlyphClosed, ttGlyphOpened),
    (ttHotGlyphClosed, ttHotGlyphOpened)
  );
var
  Details: TThemedElementDetails;
  R: TRect;
  Collapse: boolean;
begin
  ...
  //draw a themed expand sign.
  Details := ThemeServices.GetElementDetails(PlusMinusDetail[False, Collapse]);
  R := Rect(ALeft, ATop, ARight + 1, ABottom + 1);
  ThemeServices.DrawElement(Canvas.Handle, Details, R, nil);
  ...
end;

```

翻译 : <http://wiki.freepascal.org/Colors>



山东世联环保科技开发有限公司

Shandong World United Environmental Protection Technology Development Co.,Ltd.

世联环保官方网站: <http://www.cnwut.com>

感谢 山东世联环保科技开发有限公司 资助 Lazarus 书籍的中文化翻译。

希望大家帮助与支持 山东世联环保科技开发有限公司 的发展,

为 世联环保 提供业务信息, 进而支持 Lazarus 中文化发展!

