Backgrounds

The bgcolor() function changes the color of the script’s background. If the script is running in `overlay = true` mode, then it will color the chart’s background.

The function’s signature is:

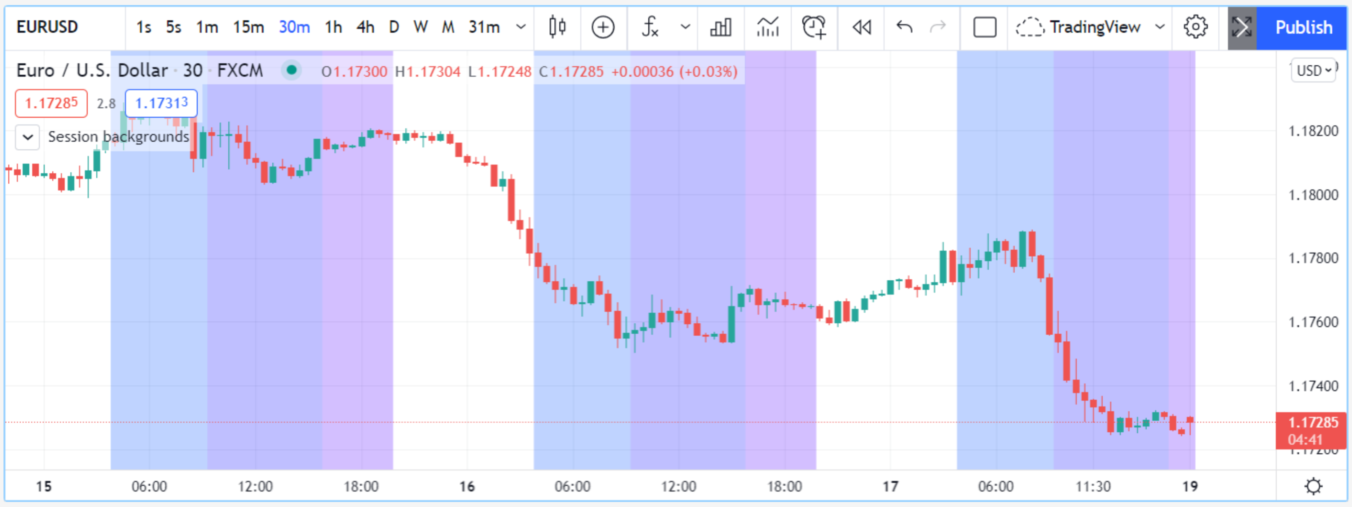
```
bgcolor(color, offset, editable, show_last, title) → void
```

Its `color` parameter allows a “series color” to be used for its argument, so it can be dynamically calculated in an expression.

If the correct transparency is not part of the color to be used, it can be be generated using the color.new() function.

Here is a script that colors the background of trading sessions (try it on 30min EURUSD, for example):

```
//@version=5
indicator("Session backgrounds", overlay = true)

// Default color constants using tranparency of 25.
BLUE_COLOR   = #0050FF40
PURPLE_COLOR = #0000FF40
PINK_COLOR   = #5000FF40
NO_COLOR     = color(na)

// Allow user to change the colors.
preMarketColor  = input.color(BLUE_COLOR, "Pre-market")
regSessionColor = input.color(PURPLE_COLOR, "Pre-market")
postMarketColor = input.color(PINK_COLOR, "Pre-market")

// Function returns `true` when the bar's time is
timeInRange(tf, session) =>
    time(tf, session) != 0

// Function prints a message at the bottom-right of the chart.
f_print(_text) =>
    var table _t = table.new(position.bottom_right, 1, 1)
    table.cell(_t, 0, 0, _text, bgcolor = color.yellow)

var chartIs30MinOrLess = timeframe.isseconds or (timeframe.isintraday and timeframe.mul
sessionColor = if chartIs30MinOrLess
    switch
        timeInRange(timeframe.period, "0400-0930") => preMarketColor
        timeInRange(timeframe.period, "0930-1600") => regSessionColor
        timeInRange(timeframe.period, "1600-2000") => postMarketColor
        => NO_COLOR
else
    f_print("No background is displayed.\nChart timeframe must be <= 30min.")
    NO_COLOR

bgcolor(sessionColor)
```



Note that:

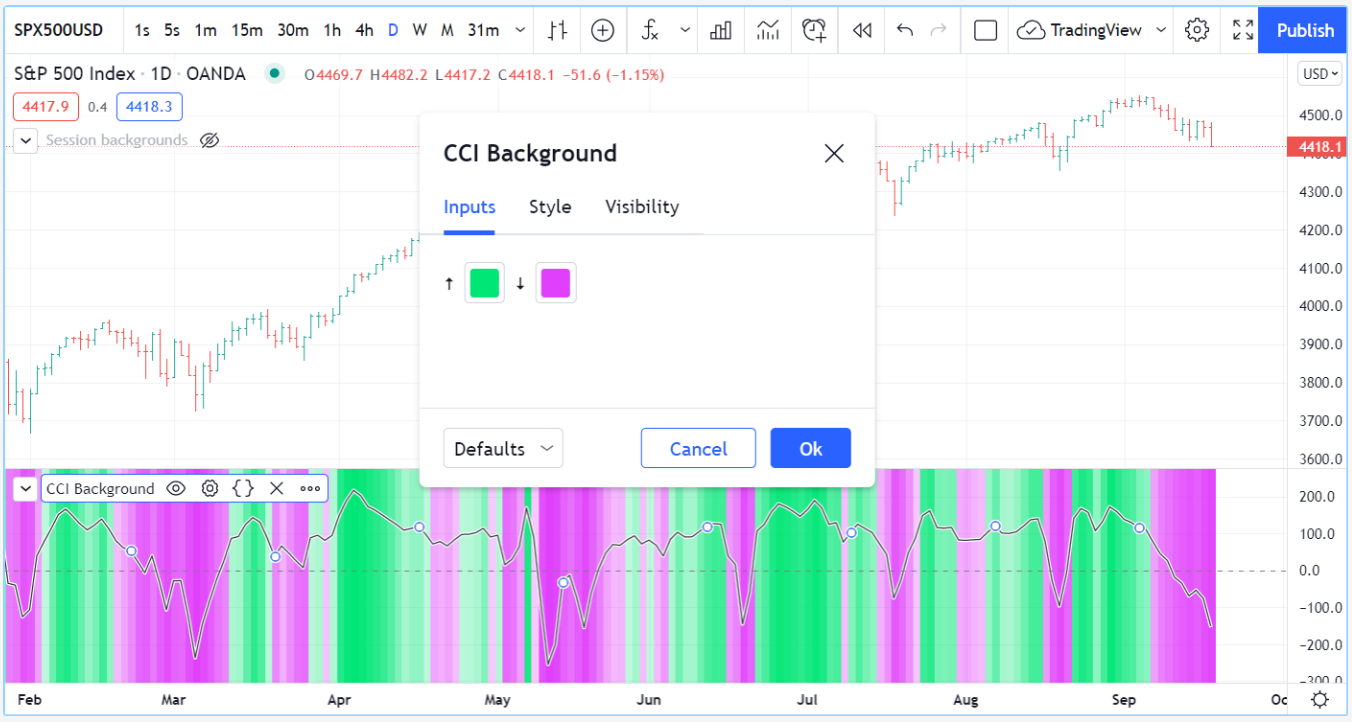
- The script only works on chart timeframes of 30min or less. It prints an error message when the chart's timeframe is higher than 30min.
- When the if structure's `else` branch is used because the chart's timeframe is incorrect, the local block returns the `NO_COLOR` color so that no background is displayed in that case.
- We first initialize constants using our base colors, which include the `40` transparency in hex notation at the end. 40 in the hexadecimal notation on the reversed 00-FF scale for transparency corresponds to 75 in Pine

Script™'s 0-100 decimal scale for transparency.

- We provide color inputs allowing script users to change the default colors we propose.

In our next example, we generate a gradient for the background of a CCI line:

```
//@version=5
indicator("CCI Background")

bullColor = input.color(color.lime, "🠅", inline = "1")
bearColor = input.color(color.fuchsia, "🠇", inline = "1")

// Calculate CCI.
myCCI = ta.cci(hlc3, 20)
// Get relative position of CCI in last 100 bars, on a 0-100% scale.
myCCIPosition = ta.percentrank(myCCI, 100)
// Generate a bull gradient when position is 50-100%, bear gradient when position is 0-
backgroundColor = if myCCIPosition >= 50
    color.from_gradient(myCCIPosition, 50, 100, color.new(bullColor, 75), bullColor)
else
    color.from_gradient(myCCIPosition, 0, 50, bearColor, color.new(bearColor, 75))

// Wider white line background.
plot(myCCI, "CCI", color.white, 3)
// Think black line.
plot(myCCI, "CCI", color.black, 1)
// Zero level.
hline(0)
// Gradient background.
bgcolor(backgroundColor)
```



Note that:

- We use the ta.cci() built-in function to calculate the indicator value.
- We use the ta.percentrank() built-in function to calculate `myCCIPosition`, i.e., the percentage of past `myCCI` values in the last 100 bars that are below the current value of `myCCI`.
- To calculate the gradient, we use two different calls of the color.from_gradient() built-in: one for the bull gradient when `myCCIPosition` is in the 50-100% range, which means that more past values are below its current value, and another for the bear gradient when `myCCIPosition` is in the 0-49.99% range, which means

that more past values are above it.

- We provide inputs so the user can change the bull/bear colors, and we place both color input widgets on the same line using `inline = "1"` in both input.color() calls.
- We plot the CCI signal using two plot() calls to achieve the best contrast over the busy background: the first plot is a 3-pixel wide white background, the second plot() call plots the thin, 1-pixel wide black line.

See the Colors page for more examples of backgrounds.

Alerts

Bar coloring

Options v: v5