Bar plotting

Introduction

The plotcandle() built-in function is used to plot candles. plotbar() is used to plot conventional bars.

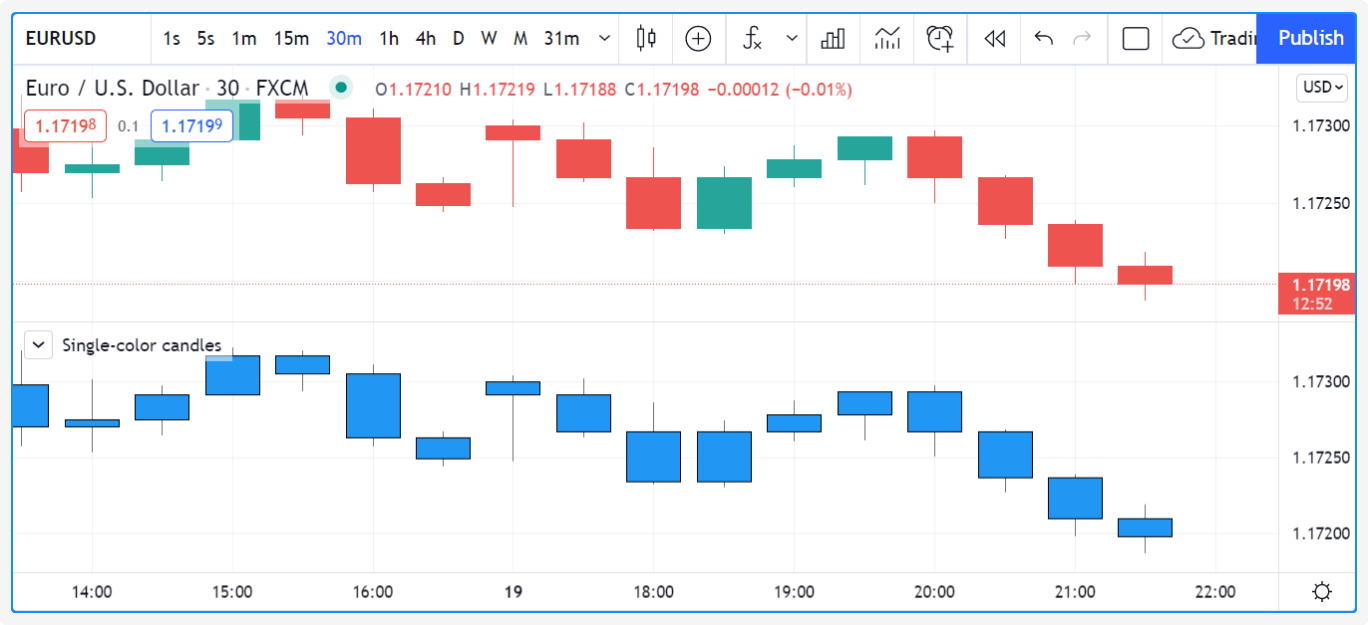
Both functions require four arguments that will be used for the OHLC prices (open, high, low, close) of the bars they will be plotting. If one of those is na, no bar is plotted.

Plotting candles with `plotcandle()`

```
plotcandle(open, high, low, close, title, color, wickcolor, editable, show_last, borde
```

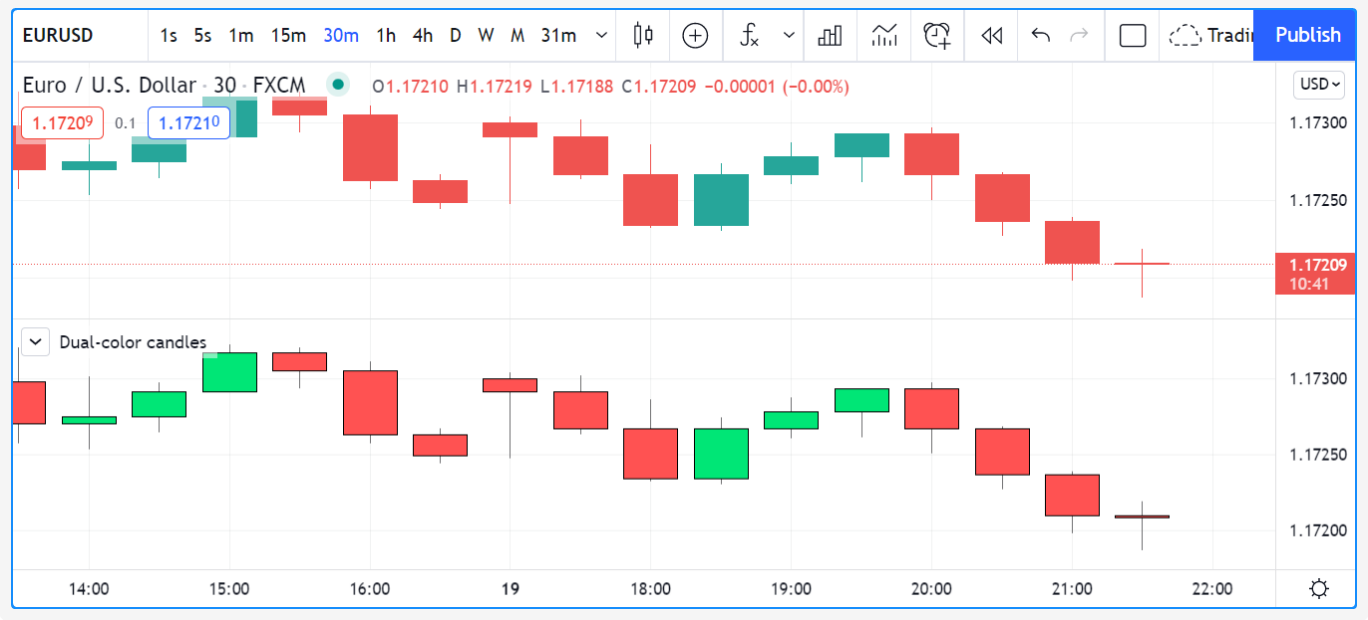
This plots simple candles, all in blue, using the habitual OHLC values, in a separate pane:

```
//@version=5
indicator("Single-color candles")
plotcandle(open, high, low, close)
```



To color them green or red, we can use the following code:

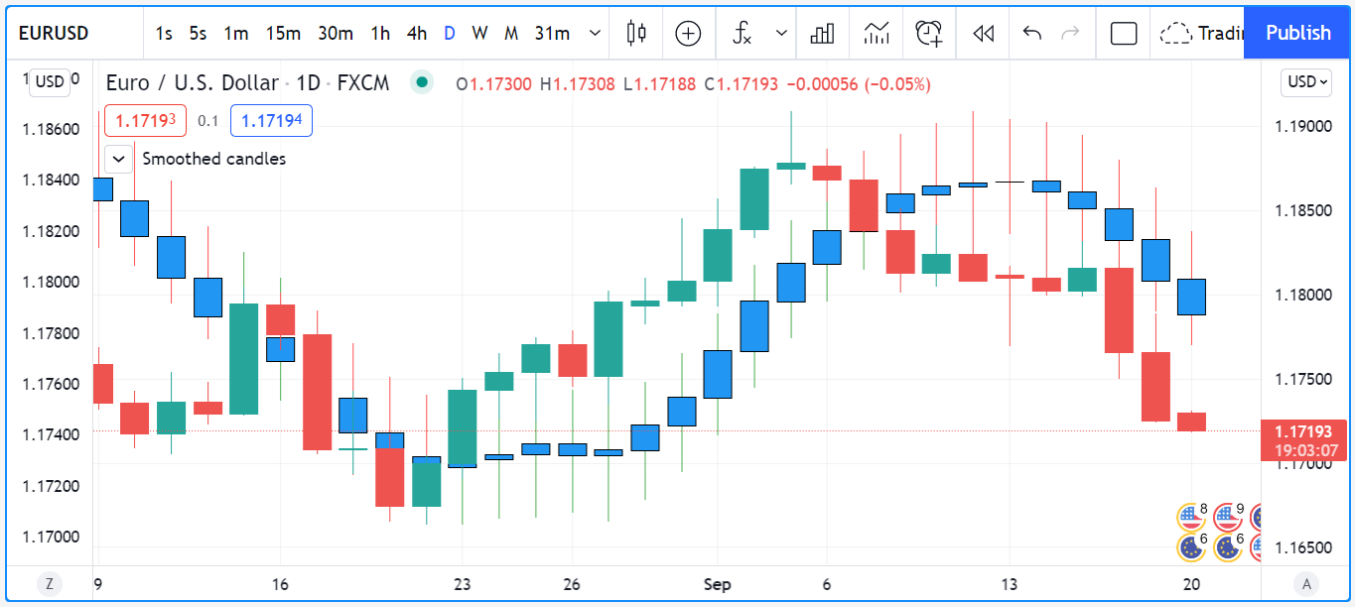
```
//@version=5
indicator("Example 2")
paletteColor = close >= open ? color.lime : color.red
plotbar(open, high, low, close, color = paletteColor)
```



Note that the `color` parameter accepts “series color” arguments, so constant values such as `color.red`, `color.lime`, `"#FF9090"`, as well as expressions that calculate colors at runtime, as is done with the `paletteColor` variable here, will all work.

You can build bars or candles using values other than the actual OHLC values. For example you could calculate and plot smoothed candles using the following code, which also colors wicks depending on the position of close relative to the smoothed close (`c`) of our indicator:

```
//@version=5
indicator("Smoothed candles", overlay = true)
lenInput = input.int(9)
smooth(source, length) =>
    ta.sma(source, length)
o = smooth(open, lenInput)
h = smooth(high, lenInput)
l = smooth(low, lenInput)
c = smooth(close, lenInput)
ourWickColor = close > c ? color.green : color.red
plotcandle(o, h, l, c, wickcolor = ourWickColor)
```

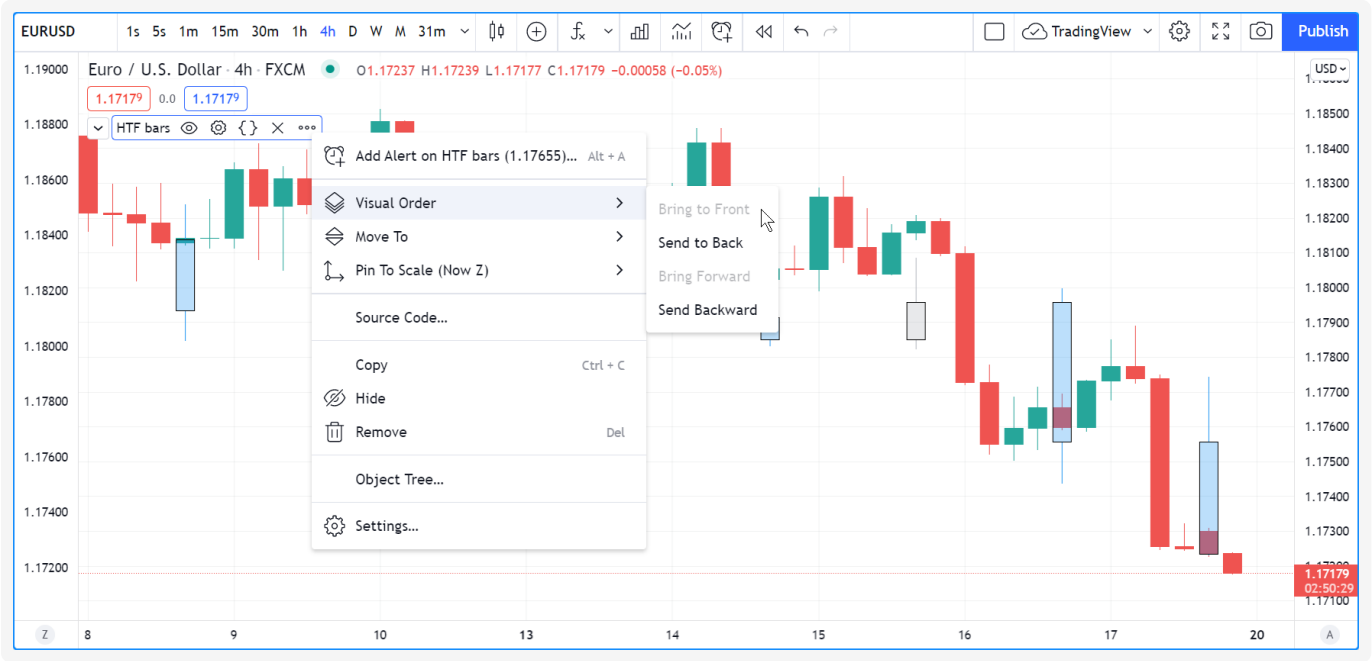


You may find it useful to plot OHLC values taken from a higher timeframe. You can, for example, plot daily bars on an intraday chart:

```
// NOTE: Use this script on an intraday chart.
//@version=5
indicator("Daily bars")

// Use gaps to only return data when the 1D timeframe completes, `na` otherwise.
[o, h, l, c] = request.security(syminfo.tickerid, "D", [open, high, low, close], gaps

var color UP_COLOR = color.silver
var color DN_COLOR = color.blue
color wickColor = c >= o ? UP_COLOR : DN_COLOR
color bodyColor = c >= o ? color.new(UP_COLOR, 70) : color.new(DN_COLOR, 70)
// Only plot candles on intraday timeframes,
// and when non `na` values are returned by `request.security()` because a HTF has comp
plotcandle(timeframe.isintraday ? o : na, h, l, c, color = bodyColor, wickcolor = wick
```



Note that:

- We show the script's plot after having used "Visual Order/Bring to Front" from the script's "More" menu. This causes our script's candles to appear on top of the chart's candles.
- The script will only display candles when two conditions are met:
    - The chart is using an intraday timeframe (see the check on `timeframe.isintraday` in the plotcandle() call). We do this because it's not useful to show a daily value on timeframes higher or equal to 1D.
    - The request.security() function returns non na values (see `gaps = barmerge.gaps_on` in the function call).
- We use a tuple (`[open, high, low, close]`) with request.security() to fetch four values in one call.
- We use var to declare our `UP_COLOR` and `DN_COLOR` color constants on bar zero only. We use constants because those colors are used in more than one place in our code. This way, if we need to change them, we need only do so in one place.
- We create a lighter transparency for the body of our candles in the `bodyColor` variable initialization, so they don't obstruct the chart's candles.

Plotting bars with `plotbar()`

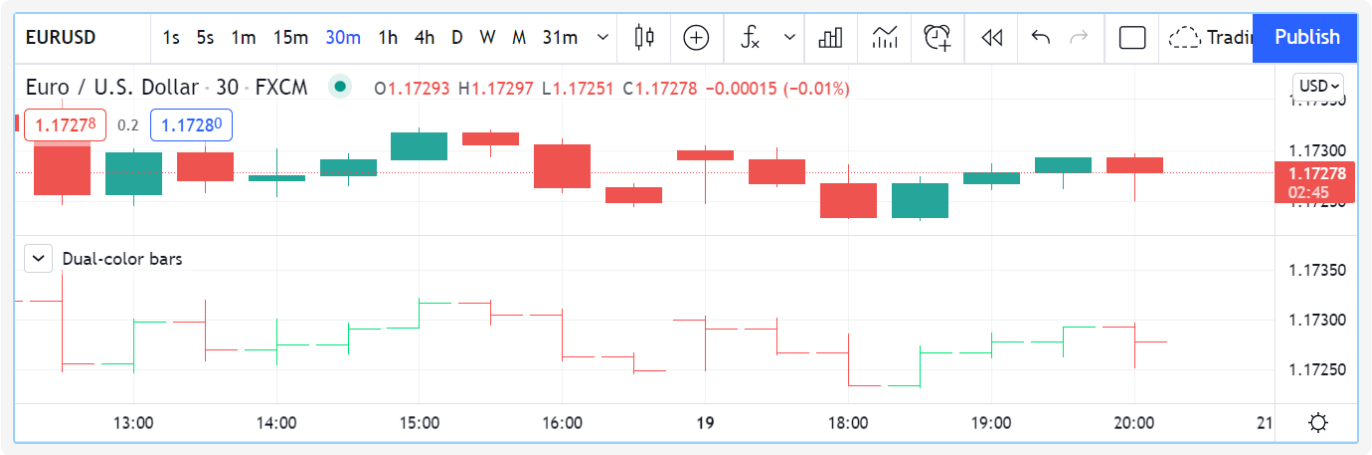
The signature of plotbar() is:

```
plotbar(open, high, low, close, title, color, editable, show_last, display) → void
```

Note that plotbar() has no parameter for `bordercolor` or `wickcolor`, as there are no borders or wicks on conventional bars.

This plots conventional bars using the same coloring logic as in the second example of the previous section:

```
//@version=5
indicator("Dual-color bars")
paletteColor = close >= open ? color.lime : color.red
plotbar(open, high, low, close, color = paletteColor)
```



TradingView

Bar coloring

Bar states