To Pine Script™ version 3

This document helps to migrate Pine Script™ code from `@version=2` to `@version=3`.

Default behaviour of security function has changed

Let's look at the simple `security` function use case. Add this indicator on an intraday chart:

```
// Add this indicator on an intraday (e.g., 30 minutes) chart
//@version=2
study("My Script", overlay=true)
s = security(tickerid, 'D', high, false)
plot(s)
```

This indicator is calculated based on historical data and looks somewhat *into the future*. At the first bar of every session an indicator plots the high price of the entire day. This could be useful in some cases for analysis, but doesn't work for backtesting strategies.

We worked on this and made changes in Pine Script™ version 3. If this indicator is compiled with `//@version=3` directive, we get a completely different picture:

The old behaviour is still available though. We added a parameter to the `security` function (the fifth one) called `lookahead`.

It can take on the form of two different values: `barmerge.lookahead_off` (and this is the default for Pine Script™ version 3) or `barmerge.lookahead_on` (which is the default for Pine Script™ version 2).

Self-referenced variables are removed

Pine Script™ version 2 pieces of code, containing a self-referencing variable:

```
//@version=2
//...
s = nz(s[1]) + close
```

Compiling this piece of code with Pine Script™ version 3 will give you an `Undeclared identifier 's'` error. It should be rewritten as:

```
//@version=3
//...
s = 0.0
s := nz(s[1]) + close
```

`s` is now a *mutable variable* that is initialized at line 3. At line 3 the initial value gives the Pine Script™ compiler the information about the variable type. It's a float in this example.

In some cases you may initialize that mutable variable (like `s`) with a `na` value. But in complex cases that won't work.

Forward-referenced variables are removed

```
//@version=2
//...
d = nz(f[1])
e = d + 1
f = e + close
```

In this example `f` is a forward-referencing variable, because it's referenced at line 3 before it was declared and initialized. In Pine Script™ version 3 this will give you an error `Undeclared identifier 'f'`. This example should be rewritten in Pine Script™ version 3 as follows:

```
//@version=3
//...
f = 0.0
d = nz(f[1])
e = d + 1
f := e + close
```

Resolving a problem with a mutable variable in a security expression

When you migrate script to version 3 it's possible that after removing self-referencing and forward-referencing variables the Pine Script™ compiler will give you an error:

```
//@version=3
//...
s = 0.0
s := nz(s[1]) + close
t = security(tickerid, period, s)
```

`Cannot use mutable variable as an argument for security function!`

This limitation exists since mutable variables were introduced in Pine Script™, i.e., in version 2. It can be resolved as before: wrap the code with a mutable variable in a function:

```
//@version=3
//...
calcS() =>
    s = 0.0
    s := nz(s[1]) + close
t = security(tickerid, period, calcS())
```

Math operations with booleans are forbidden

In Pine Script™ v2 there were rules of implicit conversion of booleans into numeric types. In v3 this is forbidden. There is a conversion of numeric types into booleans instead (0 and `na` values are `false`, all the other numbers are `true`). Example (In v2 this code compiles fine):

```
//@version=2
study("My Script")
s = close >= open
s1 = close[1] >= open[1]
s2 = close[2] >= open[2]
sum = s + s1 + s2
col = sum == 1 ? white : sum == 2 ? blue : sum == 3 ? red : na
bgcolor(col)
```

Variables `s`, `s1` and `s2` are of *bool* type. But at line 6 we add three of them and store the result in a variable `sum`. `sum` is a number, since we cannot add booleans. Booleans were implicitly converted to numbers (`true` values to `1.0` and `false` to `0.0`) and then they were added.

This approach leads to unintentional errors in more complicated scripts. That's why we no longer allow implicit conversion of booleans to numbers.

If you try to compile this example as a Pine Script™ v3 code, you'll get an error: `Cannot call `operator +` with arguments (series__bool, series__bool); <...>` It means that you cannot use the addition operator with boolean values. To make this example work in Pine Script™ v3 you can do the following:

```
//@version=3
study("My Script")
bton(b) =>
    b ? 1 : 0
s = close >= open
s1 = close[1] >= open[1]
s2 = close[2] >= open[2]
sum = bton(s) + bton(s1) + bton(s2)
col = sum == 1 ? white : sum == 2 ? blue : sum == 3 ? red : na
bgcolor(col)
```

Function `bton` (abbreviation of boolean-to-number) explicitly converts any boolean value to a number if you really need this.

To Pine Script™ version 4

Where can I get more information?

© Copyright 2023, TradingView.