3 ways to live-edit your UI

How to save time when making a Dplug plug-in UI?
Discover 3 workflow loops for your product.

Meeting
Apr 5th 2022

Visual feedback loops are important.

- To try things out quickly, get out of local optimum.
- In software, to avoid rebuild times.
- Can’t judge what you can’t see!
- Nearer to the user point of view

Visual memory “buffer” is short

When a stimulus is presented, its sensory trace decays rapidly, lasting for approximately 1000 ms. This brief and labile memory, referred as ***iconic memory****, serves as a buffer before information is transferred to working memory and executive control. -*

Graziano - Sigman (2008)

How can Dplug help?

Here are three ways to live-edit your plug-in UI.

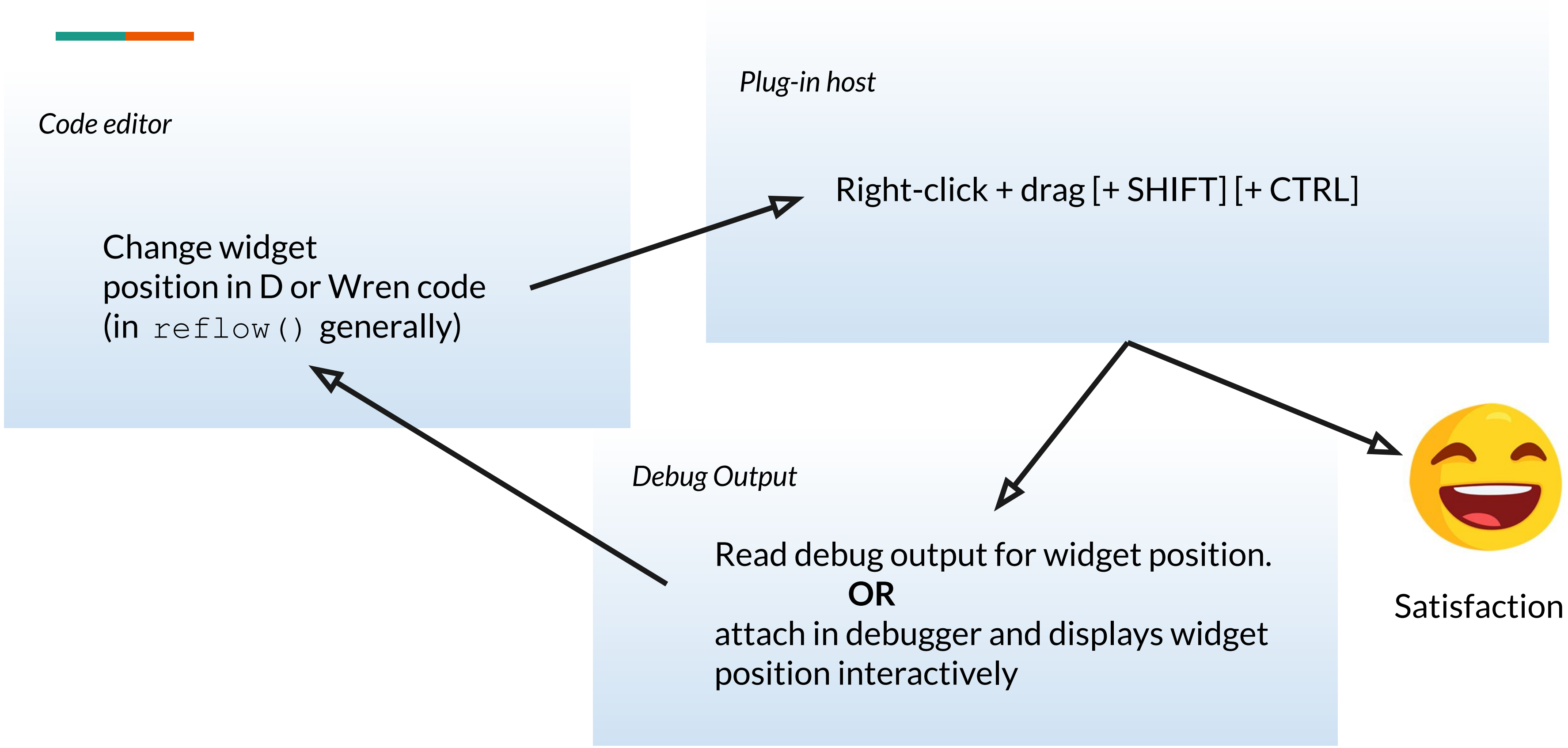
1. Right-click + drag

4 ways to do the right clic drag.

Right-clic + Drag	=> move a widget
Right-clic + SHIFT + Drag	=> resize a widget
Right-clic + Drag + CTRL	=> move a widget, per one user pixel increment
Right-clic + SHIFT + Drag + CTRL	=> resize a widget, per one user pixel increment

1. Right-clic + drag

Workflow:



Code editor

Change widget position in D or Wren code (in `reflow()` generally)

Plug-in host

Right-click + drag [+ SHIFT] [+ CTRL]

Debug Output

Read debug output for widget position.
OR
attach in debugger and displays widget position interactively

Satisfaction

1. Right-clic + drag

Workflow:

Code editor

Change widget position in D or Wren code (in `reflow()` generally)

Plug-in host

Right-click + drag [+ SHIFT] [+ CTRL]

Debug Output

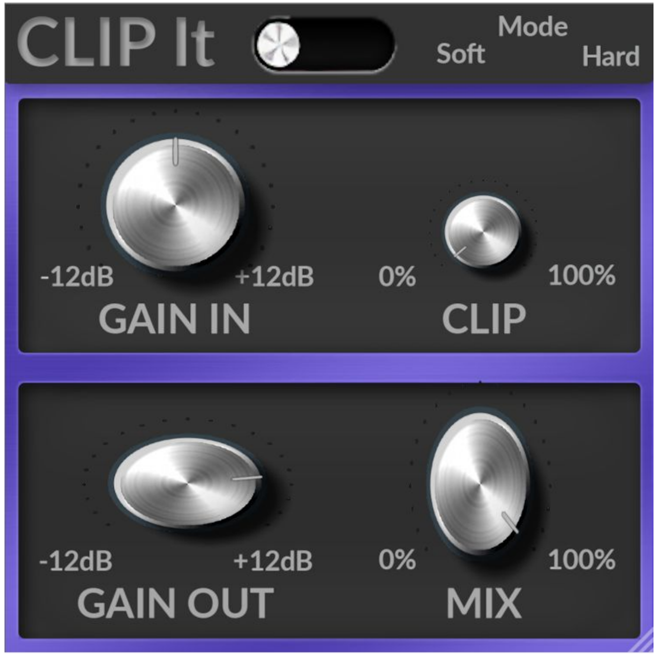
Read debug output for widget position.
OR
attach in debugger and displays widget position interactively

Satisfaction

Warning = widget rectangle at runtime is in “user pixels” (resized plug-in) but you probably want the position in scale = 1.0

1. Right-clic + drag

Demo with Good Ol' Clip It



3 Key Facts about right-clic + drag

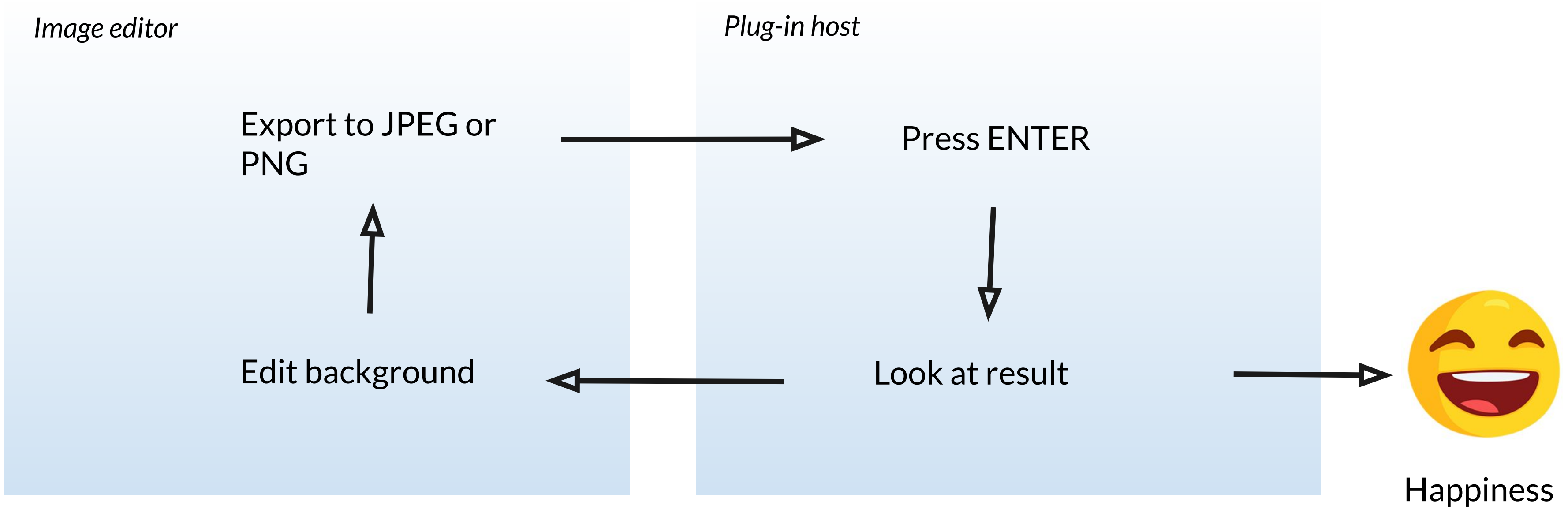
A. **Nothing to do!** This is enabled when `–debug` is used in the D compiler (DUB default to this).

B. **Read the widget position in Debug Output** (else it would be for nothing)
Windows: use DbgView.exe in Admin mode, or launch from Visual Studio
POSIX: launch the host from command-line *=> convert that position to original scale (x1.0)*

C. **How To Disable =** this is implemented in `dplug.gui.element.UIElement`
=>replace the `debug` clause by `if(false)`

2. Background reload with ENTER

Workflow:



Pro-tips for background reload

A. Need setup. Provide an **absolute path** for images.

eg: POSIX => "`/home/myuser/my/path/to/pluginname/gfx/`"
Windows => "`c:\users\myuser\my\path\to\pluginname\gfx\`"

B. Reload is enabled when `-debug` is used in the D compiler *(eg: DUB default configuration)*

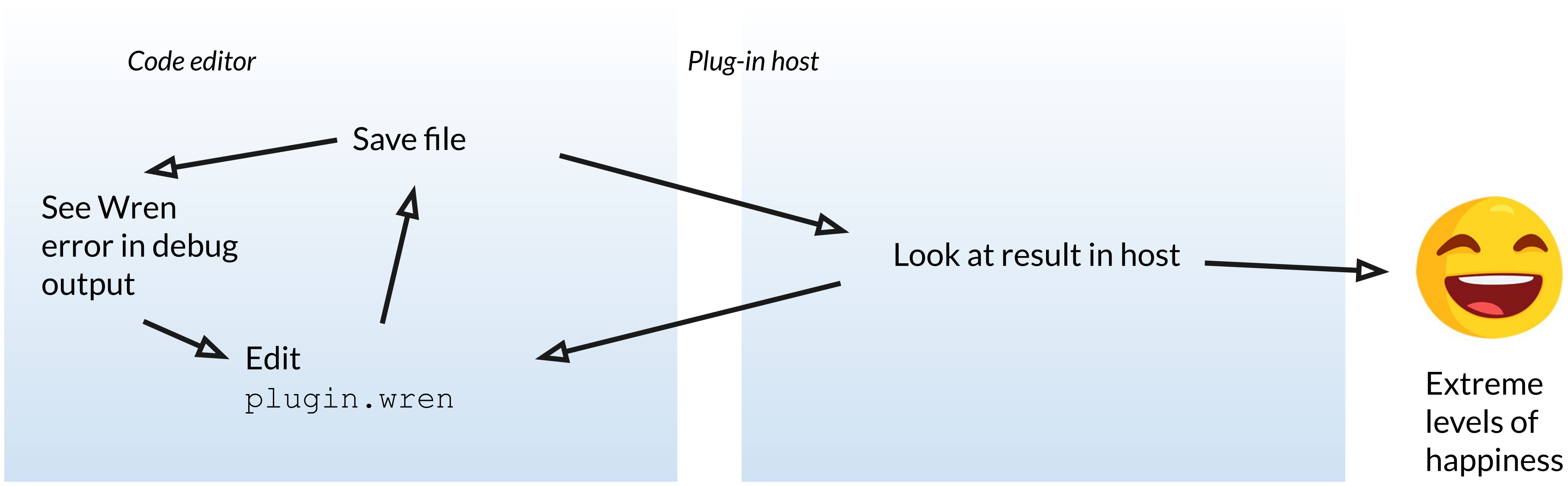
(so you can let the absolute path leak in the released build)

C. How to disable = this is implemented in `PBRBackgroundGUI` and `FlatBackgroundGUI`

In FlatBackgroundGUI since yesterday actually => Dplug v12.4.6+

3. The Wren workflow

Setup already explained in Dplug Tutorials 1 - Live-coding plugins UI with Wren and Wiki.

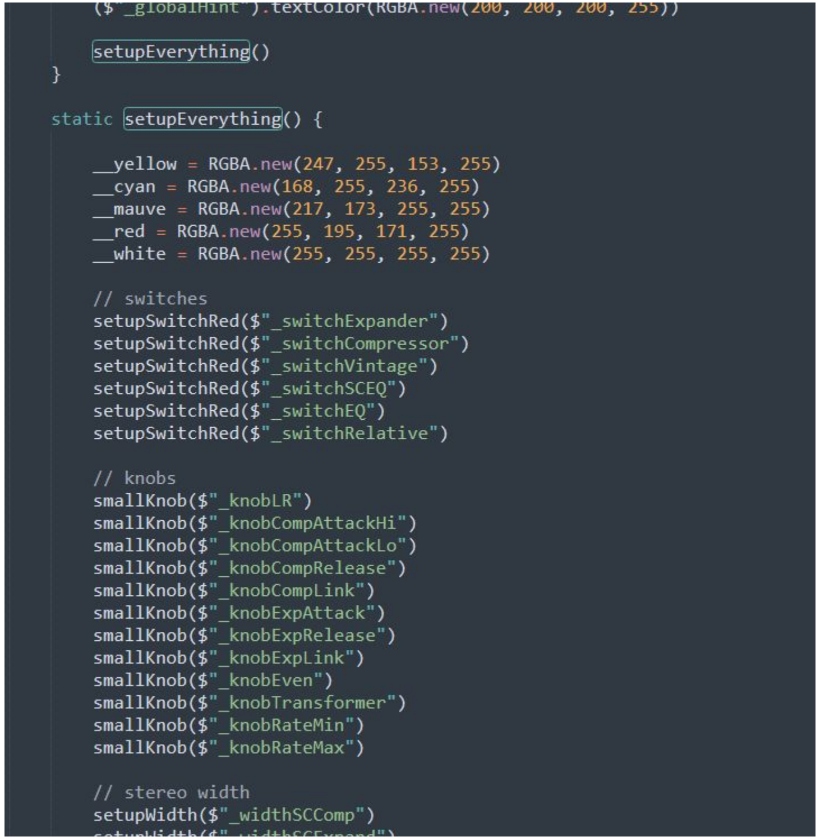


Wren tips

Tip 1: Put positioning in `reflow()`, and all other properties in a `static setupEverything()` function.

Call that from both `create()` and `reflow()`

=> because you want to setup any kind of properties in live-edit, but only reflow is called when changing a script.



```
        ($"_globalHint").textColor(RGBA.new(200, 200, 200, 255))

        setupEverything()
    }

    static setupEverything() {

        __yellow = RGBA.new(247, 255, 153, 255)
        __cyan = RGBA.new(168, 255, 236, 255)
        __mauve = RGBA.new(217, 173, 255, 255)
        __red = RGBA.new(255, 195, 171, 255)
        __white = RGBA.new(255, 255, 255, 255)

        // switches
        setupSwitchRed($"_switchExpander")
        setupSwitchRed($"_switchCompressor")
        setupSwitchRed($"_switchVintage")
        setupSwitchRed($"_switchSCEQ")
        setupSwitchRed($"_switchEQ")
        setupSwitchRed($"_switchRelative")

        // knobs
        smallKnob($"_knobLR")
        smallKnob($"_knobCompAttackHi")
        smallKnob($"_knobCompAttackLo")
        smallKnob($"_knobCompRelease")
        smallKnob($"_knobCompLink")
        smallKnob($"_knobExpAttack")
        smallKnob($"_knobExpRelease")
        smallKnob($"_knobExpLink")
        smallKnob($"_knobEven")
        smallKnob($"_knobTransformer")
        smallKnob($"_knobRateMin")
        smallKnob($"_knobRateMax")

        // stereo width
        setupWidth($"_widthSCComp")
        setupWidth($"_widthSCExpand")
```

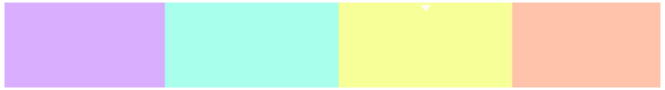
Wren tips: color scheme

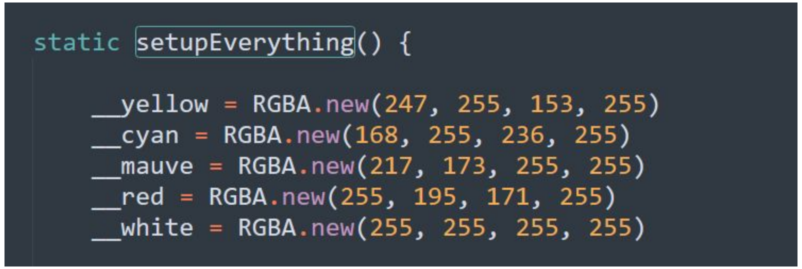
Tip 2: in `setupEverything()`, use Wren static fields to have important colors as constants (eg: `__yellow`)

You can use them in other static functions, and **have a consistent color scheme!**

Note: __yellow.withAlpha(128) is a derived RGBA color, with opacity 0.5

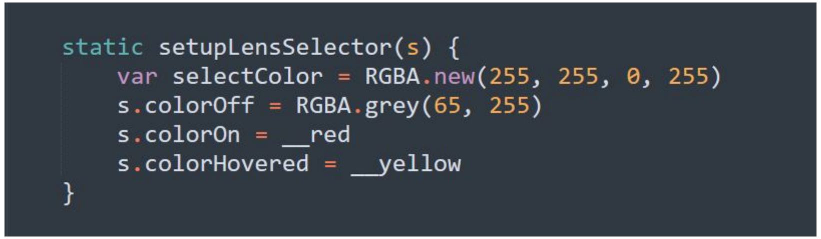
Why this tip? For some reason, creating the static fields in `create()` *does not work reliably.*





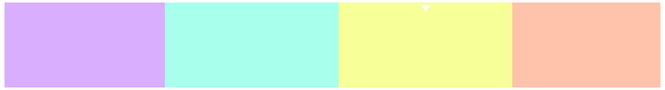
```
static setupEverything() {

    __yellow = RGBA.new(247, 255, 153, 255)
    __cyan = RGBA.new(168, 255, 236, 255)
    __mauve = RGBA.new(217, 173, 255, 255)
    __red = RGBA.new(255, 195, 171, 255)
    __white = RGBA.new(255, 255, 255, 255)
```

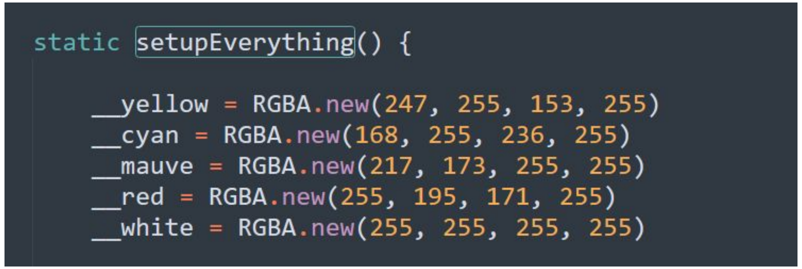


```
static setupLensSelector(s) {
    var selectColor = RGBA.new(255, 255, 0, 255)
    s.colorOff = RGBA.grey(65, 255)
    s.colorOn = __red
    s.colorHovered = __yellow
}
```

Wren tips: color scheme



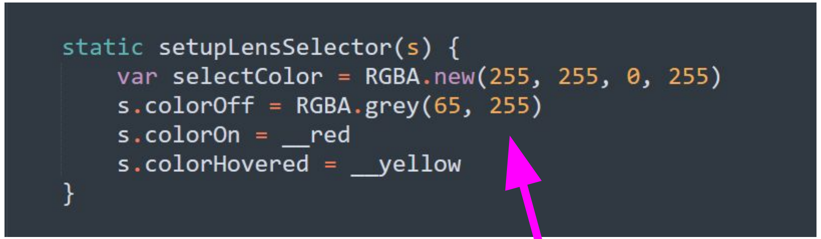
Tip 2: in `setupEverything()`, use Wren static fields to have important colors as constants (eg: `__yellow`)



```
static setupEverything() {

    __yellow = RGBA.new(247, 255, 153, 255)
    __cyan = RGBA.new(168, 255, 236, 255)
    __mauve = RGBA.new(217, 173, 255, 255)
    __red = RGBA.new(255, 195, 171, 255)
    __white = RGBA.new(255, 255, 255, 255)
```

You can use them in other static functions, and **have a consistent color scheme!**



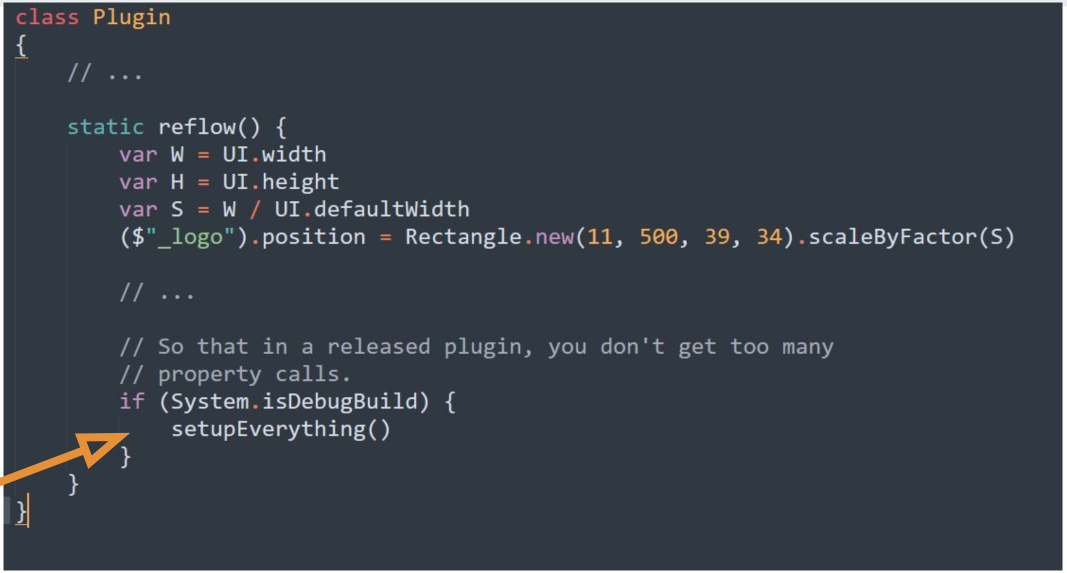
```
static setupLensSelector(s) {
    var selectColor = RGBA.new(255, 255, 0, 255)
    s.colorOff = RGBA.grey(65, 255)
    s.colorOn = __red
    s.colorHovered = __yellow
}
```

Note: _yellow.withAlpha(128) is a derived RGBA color, with opacity 0.5

Why this tip? For some reason, creating the static fields in `create()` does not work reliably.

BONUS TIP Start with grey colors! At UI beginning, you can make grey colors with `RGBA.grey(value, alpha)`

Wren tips



```
class Plugin
{
    // ...

    static reflow() {
        var W = UI.width
        var H = UI.height
        var S = W / UI.defaultWidth
        ($"_logo").position = Rectangle.new(11, 500, 39, 34).scaleByFactor(S)

        // ...

        // So that in a released plugin, you don't get too many
        // property calls.
        if (System.isDebugBuild) {
            setupEverything()
        }
    }
}
```

Tip 3: in `reflow()`, call `setupEverything()` if the plug-in is built with `–debug`

This save CPU for final plug-in, since no property needs to be set outside of `create()` for a final plugin, and setting a property invalidates the widget.

Note: Needs `wren-port` v1.1.2+

3. Wren workflow

Demo: `plugin.wren`
of a future plug-in

300 loc

Questions?