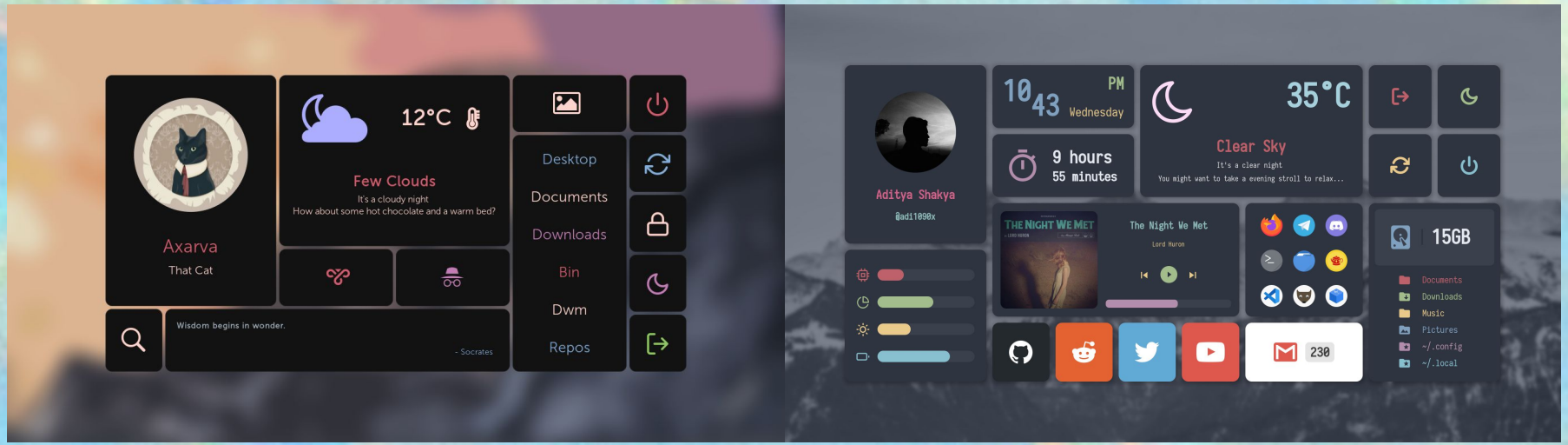
Making Widgets with Eww

brought to you by Thomas Roman

What is Eww?

- ElKowar's Wacky Widgets is a widget system made in Rust
- Eww uses Yuck for defining the widgets basic properties and CSS for styling them.
- Widget examples (from the eww github):



Installation

- First, you will need rustup to build Eww, you can install rustup with this command:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

- Next, clone and build the eww repository

```
git clone https://github.com/elkowar/eww
```

```
cd eww
```

- Build eww with one of these commands, depending on whether you are on x11 or wayland

```
cargo build --release --no-default-features --features x11
```

```
cargo build --release --no-default-features --features=wayland
```

How to make a widget!

- Make an eww directory in your .config folder to store your widgets
- Create 3 files:
  - eww.yuck - for defining windows
  - eww_widgets.yuck - for defining widgets
  - eww.scss - for styling eww

Make a window

First I will make sure to include “eww_widgets.yuck” in my “eww.yuck” file.

Next I will define a window by naming it and giving it these properties:

- Monitor - the monitor the window will be displayed on
- Stacking - whether the window will appear in the foreground (fg) or background (bg)
- Geometry - the position and size of the window, relative to the anchor.
- (test) - refers to a widget with that name which we will define in eww_widgets.yuck

```
(include "eww_widgets.yuck")

(defwindow test
  :monitor '[0]'
  :stacking "fg"
  :geometry (geometry
    :x "50"
    :y "50"
    :width "100"
    :height "100"
    :anchor "bottom right")
  (test)
)
```

Make a widget

- Now in my eww_widgets.yuck file, I will define a widget with the same name as its window.
- Inside the widget, we have many types of behaviour we can choose from
- I will be using a box and inside the box, I will add a label!
- A full list of widget behaviours is available here: https://elkowar.github.io/eww/widgets.html
- To see your widget, run the command “eww open test”
- Eww will use your GTK theme by default, so this widget may look different for you

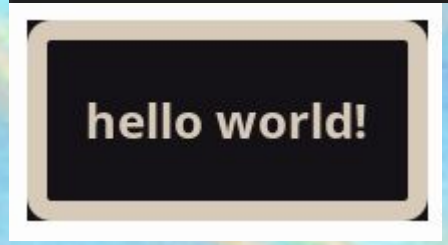
```
(defwidget test []
  (box
    (label :text "hello world!")
  )
)
```

Styling the widget with CSS

- First, we will need to add some classes to our widgets
  - We can do this using the “class” property
- Next, we can reference those classes in our eww.scss file
- Here we will define properties for the .container and .label classes
  - I also went back and adjusted the width of our widget’s window in the eww.yuck file

```
(defwidget test []
  (box :class "container"
    (label :text "hello world!" :class "label")
  )
)
```

```
.container {
  background-color: #141216;
  border-width: 10px; // for that nice thick border
  border-style: solid; // required, there are lots of funny options
  border-color: #d8cab8;
  color: #d8cab8; // in this case, this will set the font color
  border-radius: 12px; // for round corners
}
.label {
  font-size: 24px;
  font-weight: bold;
  padding-left: 12px; // it is good to set padding
  padding-right: 12px; // in case our text wraps
}
```



don’t worry, we will fix the corners on the next slide :)

Further styling

- We also need to set the background property of all windows to transparent
  - this will prevent backgrounds from escaping the boundaries of our nice round widget containers
- Also, we can unset any default theming we may have. Eww will use your GTK theme by default.
- I also set the font family here, be sure to use a font you have installed on your system.

```
window {
  background: transparent;
}
```

```
* {
  all: unset;
  font-family: Maple Mono;
}
```

Adding logic to widgets - Fun Facts widget

- Make a new file called eww_vars.yuck
- We are going to define a fact variable here
- Next, let's make a new window

```
(defwindow facts
  :monitor '[0]'
  :stacking "fg"
  :geometry (geometry
    :x "50"
    :y "50"
    :width "400"
    :height "200"
    :anchor "bottom left")
  (facts)
)
```

```
(defvar fact "click here for a fact!")
```

- Then make a widget with a box, button, and label.
- The text on the label will be the fact from eww_vars.yuck

```
(include "eww_vars.yuck")

(defwidget facts []
  (box :class "container"
    (button :class "button"
      (label :text fact :class "label")
    )
  )
)
```

Now let's talk about Bash…

- We are going to make the button call a bash script which will overwrite the fact variable
- So first, make a scripts directory and make a file called "facts.sh" in there
- Make this script executable

```
[thomasr@RomanNumerals eww]$ chmod +x ./scripts/facts.sh
```

- Next, let's define a big list of facts and randomly select one like this:

```
#!/bin/bash
# The list of Facts
facts=("A ladybird might eat more than 5,000 insects in its lifetime!" "Fruit flies were the first
facts_length="${#facts[@]}"
random_selection=$(( RANDOM % $facts_length ))
eww update fact="${facts[$random_selection]}"
```

- I sourced these facts from National Geographic Kids: https://www.natgeokids.com/uk/discover/animals/insects/15-facts-about-bugs/
- Feel free to copy and paste the list from my github:

https://github.com/tfr8811/RomanNumerals-eww-config/blob/tutorial/scripts/facts.sh

Facts widget continues..

- Now we need to call our script from our widget
- I also gave the label some added properties so it can wrap the text

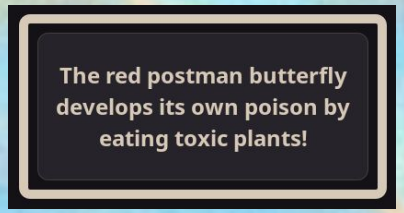
```
(defwidget facts []
  (box :class "container"
    (button
      :class "button"
      :onclick "bash ./scripts/facts.sh"
      (label
        :text fact
        :class "label"
        :wrap true
        :show-truncated false
        :wrap-mode "word"
        :justify "center"
      )
    )
  )
)
```

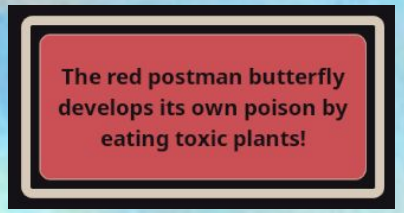
Then I will also style the “hover” state of the button like this…

```
.button {
  border-radius: 12px;
  margin: 10px;
}
```

```
.button:hover {
  background-color: #ca5056;
  color: #141216;
  transition-duration: 150ms;
}
```

```
.button:active {
  background-color: #5d252d;
  color: #141216;
  transition-duration: 150ms;
}
```



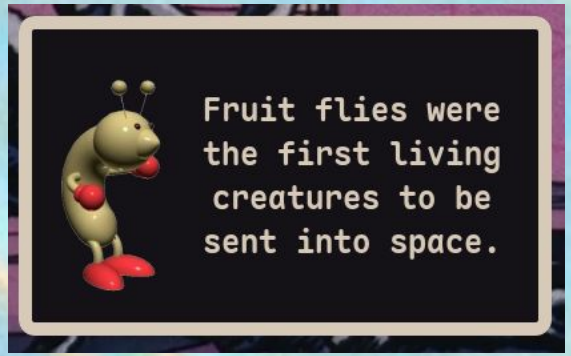


Adding images to widgets

- Make a assets folder and put a picture in there
- Add the image to your facts widget like this:

```
(defwidget facts []
  (box
  :class "container"
    (button
      :class "button"
      :onclick "bash ./scripts/facts.sh"
      (box
        :space-evenly false
        (image
        :path "./assets/cool-bug-facts.png"
        :image-width 100
        :preserve-aspect-ratio true
        )
        (label
```

←I put a box container within the button to hold both the image and the label.
I set its space-evenly property to false for the desired appearance.



Let's Make a Taskbar

- This taskbar will display the volume, music, systray, battery, time and date
- To make it we will once again, define another window and another widget
- We will also need CSS for "taskbar-container" to see our widget

```
(defwindow taskbar
  :monitor '[0]'
  :stacking "fg"
  :geometry (geometry
    :x "0"
    :y "10"
    :width "920"
    :height "40"
    :anchor "top center")
  (taskbar)
)
```

```
(defwidget taskbar []
  (box
    :class "taskbar-container"
  )
)
```

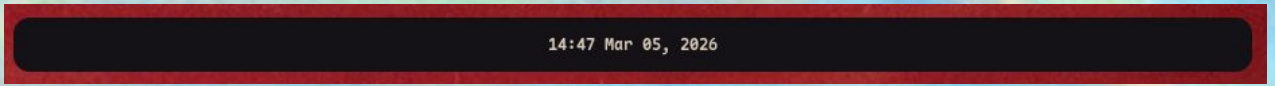
```
.taskbar-container {
  background-color: #141216;
  color: #d8cab8;
  border-radius: 12px;
}
```

Taskbar Time!

- We are going to use defpoll to set a variable called “time” at regular intervals of 10 seconds
- “A polling variable is a variable which runs a provided shell-script repeatedly, in a given interval.”

```
(defpoll time :interval "10s" "date '+%H:%M %b %d, %Y'")
```

- Next we will add a button with a label that displays the time.
- I also made a class for these labels, which we can style in the scss file.



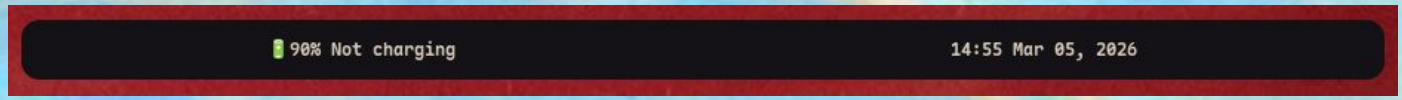

```
(defwidget taskbar []
  (box
    :class "taskbar-container"
    (button
    :class "button"
      (label
        :text "${time}"
        :class "taskbar-label"
      )
    )
```

```
.taskbar-label {
  font-size: 12px;
  font-weight: bold;
}
```

Taskbar: Battery

- To add the battery, we simply reference one of Eww's Magic Variables!!
- EWW_BATTERY has this structure: { <name>: { capacity, status } }
- You may need to check the name of your battery first, I did this by setting the text to the full EWW_BATTERY structure first.

```
(defwidget taskbar []
  (box
    :class "taskbar-container"
    (button
    :class "button"
      (label
        :text "🔋${EWW_BATTERY["BAT1"].capacity}% ${EWW_BATTERY["BAT1"].status}"
        :class "taskbar-label"
      )
    )
    (button
```



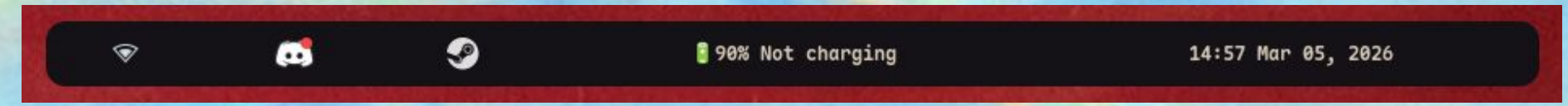
Taskbar: Systray

- Systray is another bit of functionality that comes with EWW.
- I don't put this one in a button because it already has click functionality

```
(defwidget taskbar []
  (box
    :class "taskbar-container"
    (systray
      :spacing 5
    )
    (button
```

Note: I like to remove the "all: unset;" line so that the systray menus get the GTK theme applied.

If you do this, you may need to tweak the styling of other classes.



Taskbar: Music

- Let’s get the taskbar to display the title of songs or videos we listen to.
- I added a button with a label to the widget that displays a music variable.
- I set the music variable with a deflisten
- “A listening variable runs a script once, and reads its output continously. Whenever the script outputs a new line, the value will be updated to that new line."

```
(defwidget taskbar []
  (box
    :class "taskbar-container"
    (button
    :class "button"
      (label
        :text "🎵${music}"
        :class "taskbar-label"
      )
    )
    (systray
```

```
(deflisten music :initial ""
  "playerctl --follow metadata --format '{{ artist }} - {{ title }}' || true")
```

Taskbar: Sound

- I am going to use defpoll to set a variable called “volume” at regular intervals of 1 second

```
(defpoll volume :interval "1s"
    "amixer -D pulse sget Master | awk -F '[^0-9]+' '/Left:/{print $3}'")
```

- I also grab information about the output source like this:

```
(defpoll audio-output :interval "2s"
    "pactl get-default-sink | cut -d '.' -f 1")
```

- And I display that on a button label like this:

```
(button
:class "button"
  (label
    :text "🔈${volume}% - ${audio-output}"
    :class "taskbar-label"
  )
)
```

And now we have a taskbar!!

- Possible expansions:
  - Workspaces
  - Buttons that do things
  - Dropdown menus

Using input with your widgets - notes widget

- Eww widgets can be focusable, this is useful if you want your widgets to capture input.

- Unfortunately, I couldn’t get this to work properly within the timeframe…

- Anyway, I hope to get a notes widget working sometime in the future and when I do I will share it on the RITlug Discord!

Thanks for watching and enjoy your break RITlug!