

# What's new with GStreamer & Rust

GStreamer Conference 2018

26 October 2018, Edinburgh

Sebastian 'slomo' Dröge

< [sebastian@centricular.com](mailto:sebastian@centricular.com) >

**Who?**

**What?**

# Why Rust?



Fast, explicit, zero-overhead & modern

Memory safety and thread-safety

# Status of the bindings

# What exists?

- Almost all of core, most of the libraries covered
  - Basically full-featured
  - Audio/video, pbutils, player, net, base, webrtc, sdp, rtsp, rtsp-server libraries
- Subclassing for
  - Element, Bin, Pipeline
  - Base/PushSrc, BaseTransform, BaseSink, Aggregator
  - Pad/ProxyPad/GhostPad, AggregatorPad
  - ChildProxy, UriHandler



Seriously consider Rust for your next  
GStreamer-based project

# Updates since last year

- 0.9, 0.10, 0.11 and 0.12 major releases
  - more bugfix releases
- gst-plugin-rs release
- New contributors (23+)
- New examples, example elements
- Various tutorials ported

Many new users and applications  
using the bindings

**Some code examples**

# Buffer from any Rust memory

```
// Create a 320x240 BGRx black memory
let mem = vec![0; 320*240*4];
// Fill it somehow here
let buffer = gst::Buffer::from_slice(mem);
```

# Safer time calculations

```
// Get a generic gst::Segment from somewhere and try to handle it
// as time segment. All values are in gst::ClockTime
let segment = segment.downcast_mut::<gst::ClockTime>()?;

// gst::CLOCK_TIME_NONE calculations don't wrap around
let stop = segment.get_stop() + 10 * gst::SECOND;
// stop stays NONE or is 10s higher now

// Set stop if it's smaller than duration
let dur = element.query_duration::<gst::ClockTime>()?;
if !stop.is_none() & stop < dur {
    segment.set_stop(stop);
} else {
    segment.set_stop(dur);
}
```

# Status return types

```
// Make use of Rust-style error handling via Result  
element.set_state(gst::State::Playing)  
    .into_result()?;
```

# Query/Message/Event API

```
let mut q = gst::Query::new_position(gst::Format::Time);
if !pipeline.query(&mut q) { return None; }
// Type-system knows that this is still a position query
let pos = q.get_result();

// Previously
let mut q = gst::Query::new_position(gst::Format::Time);
if !pipeline.query(q.get_mut().unwrap()) { return None; }
let pos = match q.view() {
    QueryView::Position(ref p) => p.get_result(),
    _ => unreachable!(),
};
```



**New bindings**

WebRTC

RTSP server

Discoverer and EncodingProfile (encodebin)

Metas, BufferPools, CapsFeatures

Lots of other smaller things

# **What else?**

Usability improvements & bugfixes

# gst-plugin-rs

- Lots of new base classes
- 2 HowTos, more to come
- Various new elements
  - rust-av (experimental)
  - togglerecord,  
threadshare
  - NDI
  - Example elements



Seriously consider Rust for your  
next GStreamer plugin, too!

# Success Stories

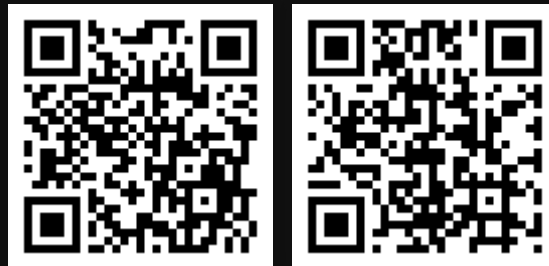
Servo: webaudio + audio/video



<https://servo.org>

# GNOME applications

- Fractal (Riot.im client)
  - <https://wiki.gnome.org/Apps/Fractal>
- Podcasts
  - <https://wiki.gnome.org/Apps/Podcasts>



Newtek NDI audio/video source



<https://github.com/teltek/gst-plugin-ndi>

glide - Cross-platform, simple video player



<https://github.com/philn/glide>

# Media TOC - Split media files into chapters



<https://github.com/fengalin/media-toc>

... and more!

Search on GitHub, crates.io, etc.



**What next?**

The bindings are basically "done"

Move to freedesktop.org GitLab  
and become part of the GStreamer project

# Bindings for the GL library

Write more applications and plugins in Rust

... and library code?

Your chance to get involved!

## Unsorted ideas

- RTSP connection/message, RTSP server
- SDP
- adaptivedemux, HLS/DASH
- HTTP server sink
- Codec parsers
- RTP
- Unit/integration tests for C components



Consider Rust instead of C in the future

# Thanks! Questions?

sebastian@centricular.com

<https://github.com/sdroege/gstreamer-rs>

<https://github.com/sdroege/gst-plugin-rs>

