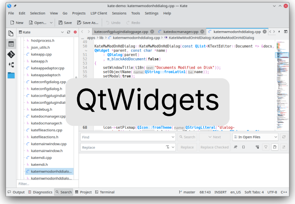
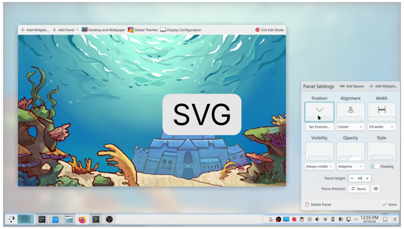
The State of (the) Union

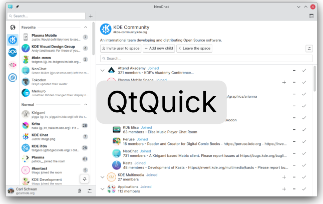
Arjen Hiemstra

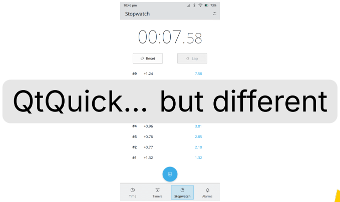
Akademy 2025

4 Applications, 4 Ways to Style

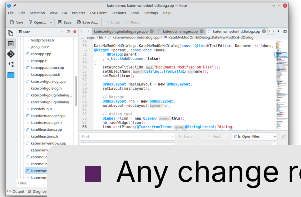


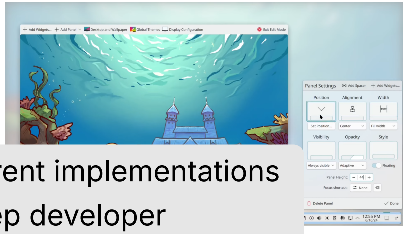


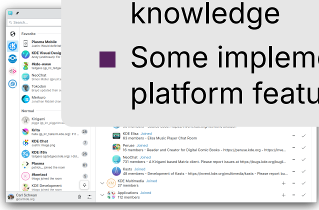




The Problems

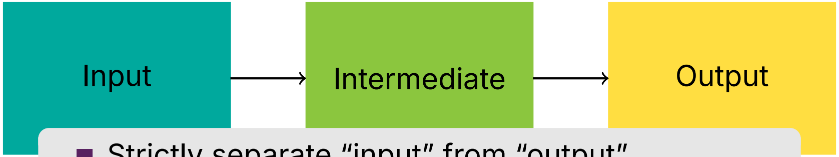






- Any change requires 4 different implementations
- Implementation requires deep developer knowledge
- Some implementations do not make full use of platform features

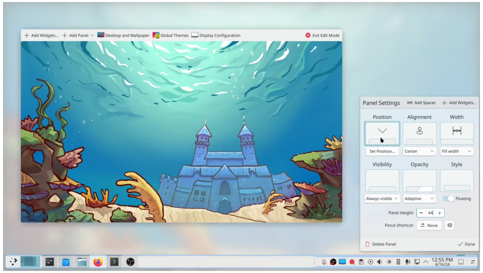
A Potential Solution



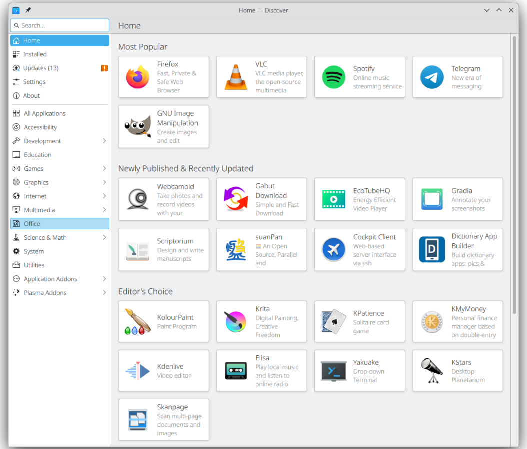
- Strictly separate “input” from “output”
- Define an “intermediate layer” common to both
- Input produces and output consumes intermediate data

Implementation Plan

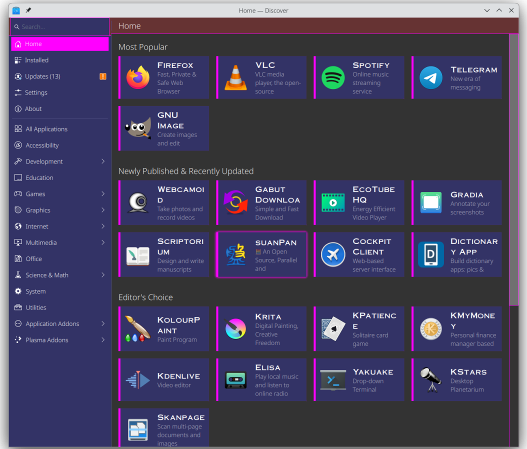
- Initial focus on a QtQuick Controls style as output
- Use Plasma's SVGs as input as it has known results
- Define the core library, then revist input/output



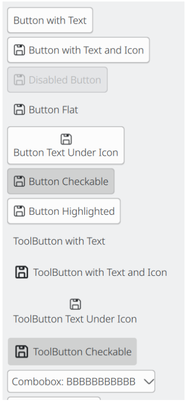
Discover


Home — Discover
Search...
Home
Installed
Updates (13)
Settings
About
All Applications
Accessibility
Development
Education
Games
Graphics
Internet
Multimedia
Office
Science & Math
System
Utilities
Application Addons
Plasma Addons
Home
Most Popular
Firefox
Fast, Private & Safe Web Browser
VLC
VLC media player, the open-source multimedia
Spotify
Online music streaming service
Telegram
New era of messaging
GNU Image Manipulation
Create images and edit
Newly Published & Recently Updated
Webcamoid
Take photos and record videos with your
Gabut Download
Simple and Fast Download
EcoTubeHQ
Energy Efficient Video Player
Gradia
Annotate your screenshots
Scriptorium
Design and write manuscripts
suanPan
An Open Source, Parallel and
Cockpit Client
Web-based server interface via ssh
Dictionary App Builder
Build dictionary apps: pics &
Editor's Choice
KolourPaint
Paint Program
Krita
Digital Painting, Creative Freedom
KPatience
Solitaire card game
KMyMoney
Personal finance manager based on double-entry
Kdenlive
Video editor
Elisa
Play local music and listen to online radio
Yakuake
Drop-down Terminal
KStars
Desktop Planetarium
Skanpage
Scan multi-page documents and images

Also Discover


Home — Discover
Search...
Home
Installed
Updates (13)
Settings
About
All Applications
Accessibility
Development
Education
Games
Graphics
Internet
Multimedia
Office
Science & Math
System
Utilities
Application Addons
Plasma Addons
Home
Most Popular
Firefox Fast, Private & Safe Web Browser
VLC VLC media player, the open-source
Spotify Online music streaming service
Telegram New era of messaging
GNU Image Create images and edit
Newly Published & Recently Updated
Webcamoid Take photos and record videos
Gabut Download Simple and Fast Download
EcoTube HQ Energy Efficient Video Player
Gradia Annotate your screenshots
Scriptorium Design and write manuscripts
SuanPan An Open Source, Parallel and
Cockpit Client Web-based server interface
Dictionary App Build dictionary apps: pics &
Editor's Choice
KolourPaint Paint Program
Krita Digital Painting, Creative Freedom
KPatience Solitaire card game
KMyMoney Personal finance manager based
Kdenlive Video editor
Elisa Play local music and listen to online radio
Yakuake Drop-down Terminal
KStars Desktop Planetarium
Skanpage Scan multi-page documents and images

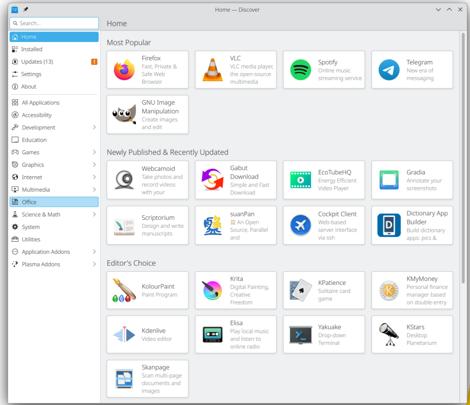
QtQuick Controls



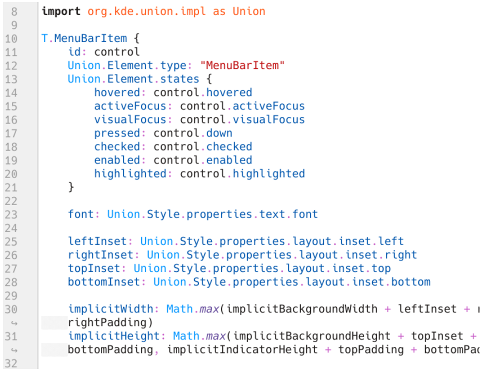
- Lots of QtQuick Controls covered
- Some Controls API details still to work out
- Several newer controls have no Breeze styling yet

Kirigami

- Kirigami integration plugin for units etc.
- Overrides to allow theming custom Kirigami controls
- Some trouble with certain controls


Home — Discover
Search...
Home
Installed
Updates (13)
Settings
About
All Applications
Accessibility
Development
Education
Games
Graphics
Internet
Multimedia
Office
Science & Math
System
Utilities
Application Addons
Plasma Addons
Home
Most Popular
Firefox Fast, Private & Safe Web Browser
VLC VLC media player, the open-source multimedia
Spotify Online music streaming service
Telegram New era of messaging
GNU Image Manipulation Create images and edit
Newly Published & Recently Updated
Webcamoid Take photos and record videos with your
Gabut Download Simple and Fast Download
EcoTubeHQ Energy Efficient Video Player
Gradia Annotate your screenshots
Scriptorium Design and write manuscripts
suanPan An Open Source, Parallel and
Cockpit Client Web-based server interface via ssh
Dictionary App Builder Build dictionary apps: pics &
Editor's Choice
KolourPaint Paint Program
Krita Digital Painting, Creative Freedom
KPatience Solitaire card game
KMyMoney Personal finance manager based on double-entry
Kdenlive Video editor
Elisa Play local music and listen to online radio
Yakuake Drop-down Terminal
KStars Desktop Planetarium
Skanpage Scan multi-page documents and images

QtQuick API details

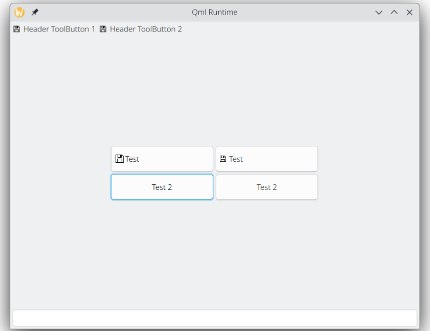


```
import org.kde.union.impl as Union

T.MenuBarItem {
    id: control
    Union.Element.type: "MenuBarItem"
    Union.Element.states {
        hovered: control.hovered
        activeFocus: control.activeFocus
        visualFocus: control.visualFocus
        pressed: control.down
        checked: control.checked
        enabled: control.enabled
        highlighted: control.highlighted
    }

    font: Union.Style.properties.text.font

    leftInset: Union.Style.properties.layout.inset.left
    rightInset: Union.Style.properties.layout.inset.right
    topInset: Union.Style.properties.layout.inset.top
    bottomInset: Union.Style.properties.layout.inset.bottom

    implicitWidth: Math.max(implicitBackgroundWidth + leftInset + 
    rightPadding)
    implicitHeight: Math.max(implicitBackgroundHeight + topInset +
    bottomPadding, implicitIndicatorHeight + topPadding + bottomPa
```

- QtQuick API working mostly through attached properties
- Custom renderer for rectangles
- Item subcontrol layout based on style information

Input Format

- Initial implementation using Plasma SVG styling
- SVG alone not enough, needed lots of extra information
- It works, but is not future proof



Requirements for a New Input Format

- No logic
- Easy to change
- Extensible to many different things

Advantages of CSS

```css
body {
    color: black;
    background-color: #fff;
}

h1 {
    font-size: 20pt;
    color: rgba(128, 0, 255, 0.9);
}
```

- It is well known
- It is very actively developed
- It is designed to abstractly describe a style

Disadvantages

```css
body {
    color: black;
    background-color: #fff;
}

h1 {
    font-size: 20pt;
    color: rgba(128, 0, 255, 0.9);
}
```

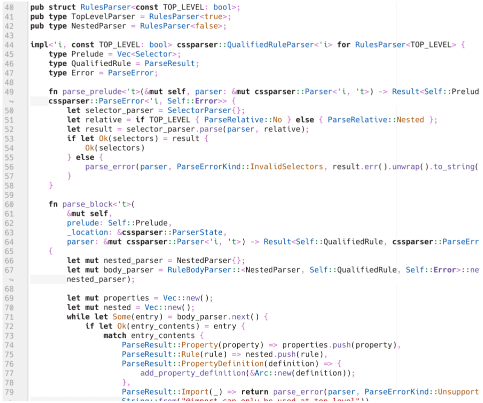
- A lot of its design is geared towards Web
- Nearly all implementations are coupled to web browsers
- Those that are not do not implement modern CSS

Enter Servo

- Servo is a web engine written in Rust
- Started by Mozilla
- Setup to be quite modular

The cssparser Crate

- A Rust crate for parsing CSS
- Used by Servo to build its CSS parser
- Implements many modern CSS features



```rust
pub struct RulesParser<const TOP_LEVEL: bool>;
pub type TopLevelParser = RulesParser<true>;
pub type NestedParser = RulesParser<false>;

impl<'i, const TOP_LEVEL: bool> cssparser::QualifiedRuleParser<'i> for RulesParser<TOP_LEVEL> {
    type Prelude = Vec<Selector>;
    type QualifiedRule = ParseResult;
    type Error = ParseError;

    fn parse_prelude<'t>(&mut self, parser: &mut cssparser::Parser<'i, 't>) -> Result<Self::Prelud
    cssparser::ParseError<'i, Self::Error>> {
        let selector_parser = SelectorParser{};
        let relative = if TOP_LEVEL { ParseRelative::No } else { ParseRelative::Nested };
        let result = selector_parser.parse(parser, relative);
        if let Ok(selectors) = result {
            Ok(selectors)
        } else {
            parse_error(parser, ParseErrorKind::InvalidSelectors, result.err().unwrap().to_string(
        }
    }

    fn parse_block<'t>(
        &mut self,
        prelude: Self::Prelude,
        _location: &cssparser::ParserState,
        parser: &mut cssparser::Parser<'i, 't>) -> Result<Self::QualifiedRule, cssparser::ParseErr
    {
        let mut nested_parser = NestedParser{};
        let mut body_parser = RuleBodyParser::<NestedParser, Self::QualifiedRule, Self::Error>::ne
        nested_parser);

        let mut properties = Vec::new();
        let mut nested = Vec::new();
        while let Some(entry) = body_parser.next() {
            if let Ok(entry_contents) = entry {
                match entry_contents {
                    ParseResult::Property(property) => properties.push(property),
                    ParseResult::Rule(rule) => nested.push(rule),
                    ParseResult::PropertyDefinition(definition) => {
                        add_property_definition(&Arc::new(definition));
                    },
                    ParseResult::Import(_) => return parse_error(parser, ParseErrorKind::Unsupport
                    String::from("@import can only be used at top level"))
```

Building a CSS Parser

- cssparser is more of a toolkit to build a CSS parser with
- Some extra helper crates for parsing selectors and colors
- No real data structures
- Quite some glue code needed for it to be useful

Integrating Rust

- Union is a C++ project
- Rust code can be integrated through helper crates like cxx
- Decided to create a stand-alone library for CSS parsing

cxx-rust-cssparser

- C++ wrapper around Rust core
- Abstract representation of a CSS file
- Designed as mostly generic library

Selectors

- Selectors define what properties to apply to which elements
- Union has the exact same concept
- Certain selectors can be directly mapped

```
body p > span::first-child {
}
nav li.active a[href] {
}
```

Selectors: Example

```
T.Button {
    Element.type: "button"
    Element.states {
        hovered: control.hovered
        pressed: control.pressed
    }
    Element.hints: ["primary"]
}
```

```
button {
}

button:hovered {
}

button.primary:pressed {
}
```

Properties

- Properties indicate what changes to make to an element
- Web's properties do not necessarily match Union
- cxx-rust-cssparser makes no assumptions about properties

```
button {
    color: red;
    background-color: black;
    padding: 4px;
}
```

“Custom” Properties

```
button {
    width: 24px;
    height: 24px;
    spacing: 4px;
    padding: 8px;
    border-radius: 4px;
}
```

- Parser needs to know about properties to parse them correctly
- Custom property syntax introduced for CSS
- Union defines all its properties using that syntax
- Match web properties where possible and it makes sense

Functions

- CSS functions can add quite useful functionality
- Unfortunately, almost all require manual implementation
- Due to this, only a handful implemented

```
--medium-element-size: 24px;

width: var(--medium-element-size);
height: var(--medium-element-size);
color: mix(#000, #fff, 0.5);
```

Conclusions

- CSS fits our usecase very well
- Implementation was not straightforward
- Future-proof solution that gives a lot of power and flexibility

First Release!

- “Tech Preview” once QtQuick Controls style is mostly complete
- Will contain CSS version of Breeze
- Should be future-proof enough to allow new development

A New Style

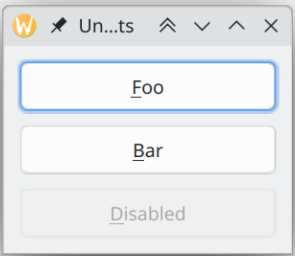
- Plasma Next has been working on a design system
- Eventually hopes to build a full application style
- Will probably need extra development on Union

Expand CSS Capabilities

- More advanced fills: Gradients, Textures
- More selectors and combinators
- Animations

More Outputs

- QtWidgets output started, but needs more work
- Potentially add integration outputs similar to Kirigami
- Other projects with styling such as decorations



Questions?

Find the code at:
https://invent.kde.org/plasma/union

Discuss things:
#union:kde.org