A Browser-Based Git Repository Visualisation Tool

https://gizual.com/

Stefan Schintler, Andreas Steinkellner, and Keith Andrews

UX Graz and CSS in Graz Joint Meetup, Tue 18 Feb 2025, 18:30

Github: https://github.com/gizual/gizual
Resources: https://gizual.com/resources

About Us

Stefan Schintler

steschi

Omnity e.U.

Andreas Steinkellner

NarrowCode

Solasit e.U.

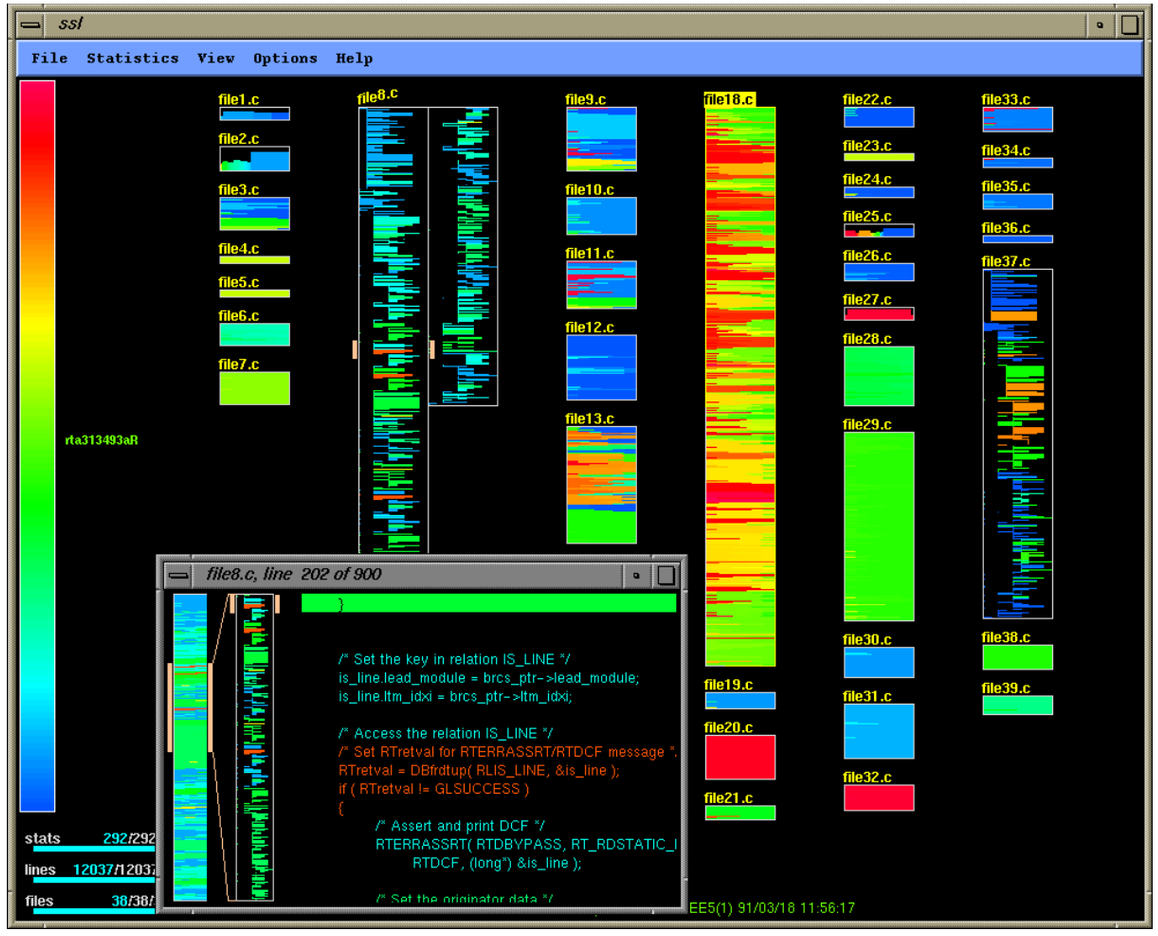
Keith Andrews

HCC, TU Graz

Background

Seesoft [1992]

- Stephen Eick et al, 1992[1].
- Visualisation of line-oriented software statistics
- Colour-coding each line of code:
  - Age (scale: blue = cold to red = hot).
  - Author (discrete palette).

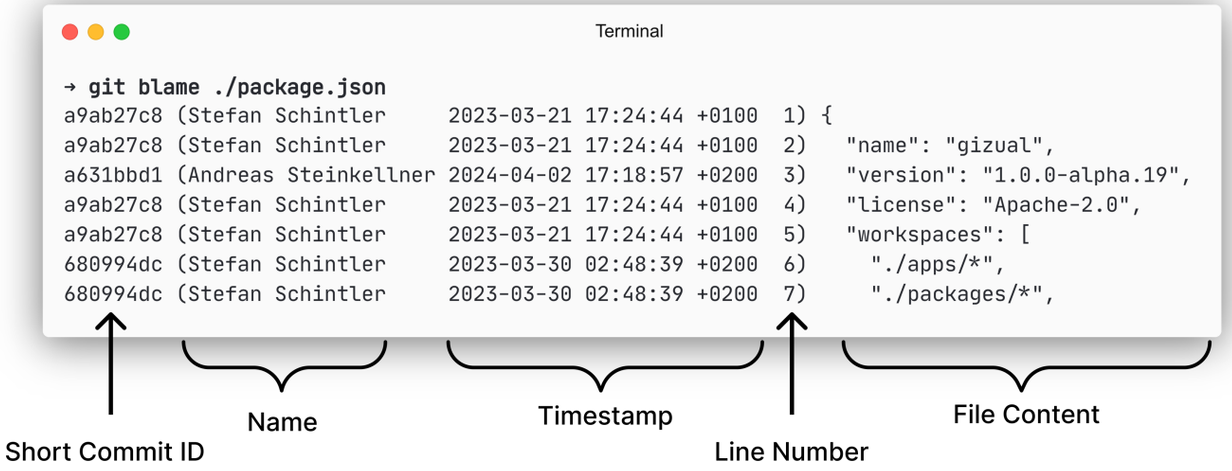


Seesoft. [Image used with kind permission of Steve Eick.]

1. *Seesoft - A Tool for Visualizing Line Oriented Software Statistics*; Stephen Eick et al; IEEE Transactions on Software Engineering; No. 18, Vol. 11, Nov 1992. doi:10.1109/32.177365

Git and Git Blame [2006]

- Git Command Line Interface[1].
- Widespread adoption.
- Cross-platform availability. (Linux, macOS, Windows)
- Dependency on POSIX shell.

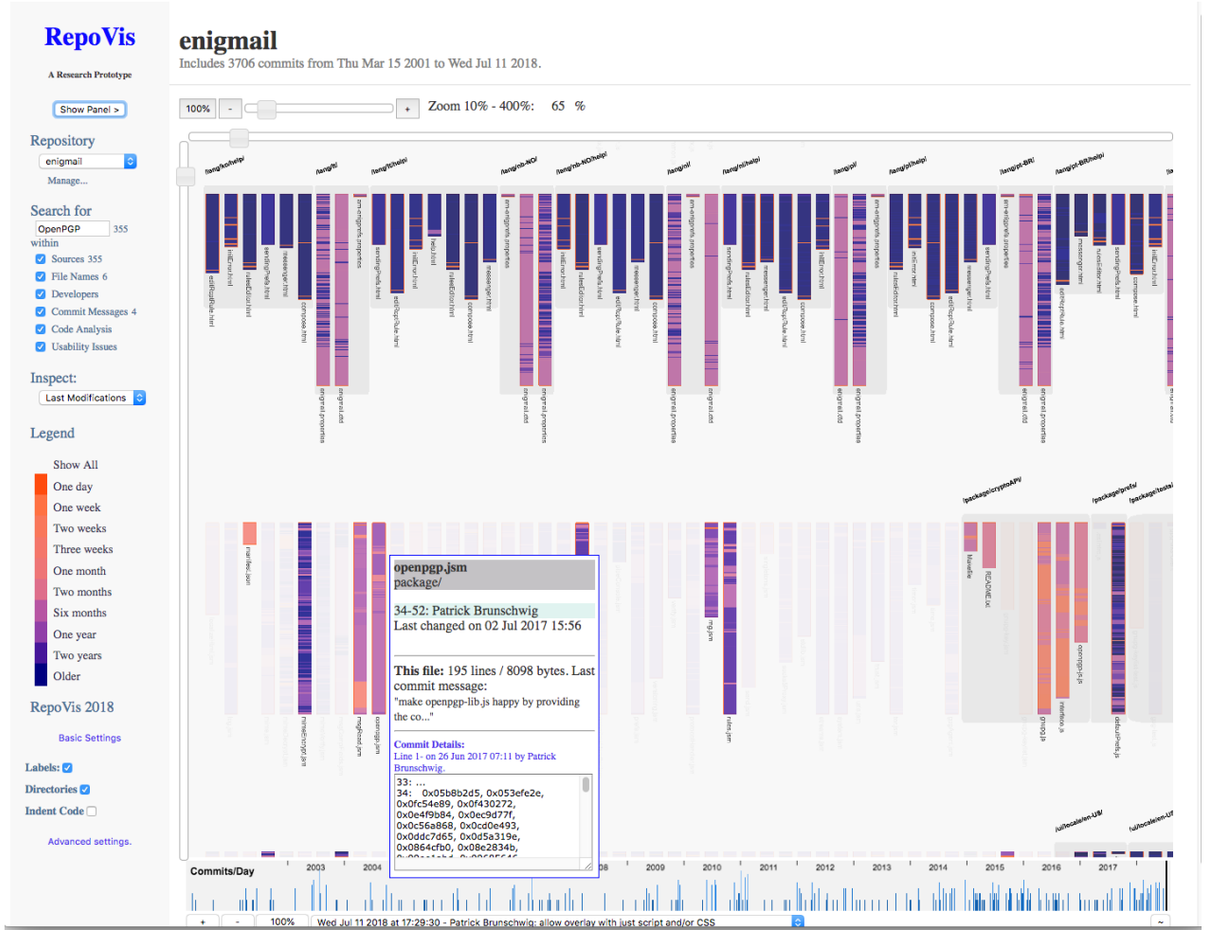


Output of Git CLI blame command.

1. Git Command Line; https://git-scm.com/

RepoVis [2018]

- Johannes Feiner and Keith Andrews[1].
- Line-oriented visualisation.
- Age and Author colour-coding.
- Backend: Rack, Ruby Sinatra, CouchDB, libgit2.
- Frontend: JavaScript, PixiJS, WebGL.



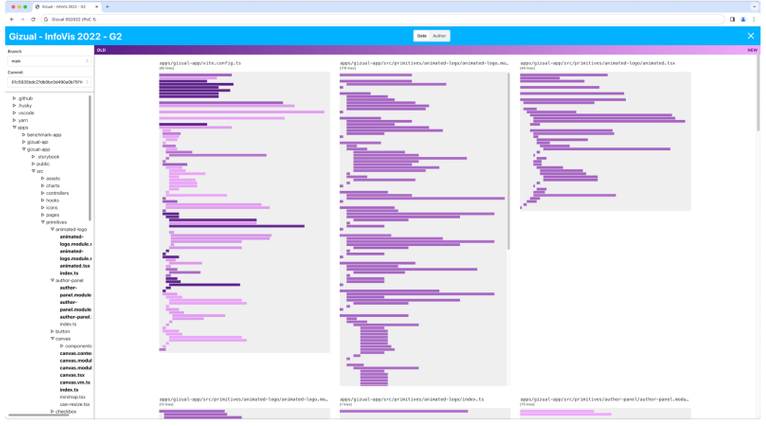
RepoVis.

1. *RepoVis: Visual Overviews and Full-Text Search in Software Repositories;* Johannes Feiner and Keith Andrews; Proc. 6th IEEE Working Conference on Software Visualization (VISSOFT 2018); Madrid, Spain, 24 Sep 2018. doi:10.1109/VISSOFT.2018.00009

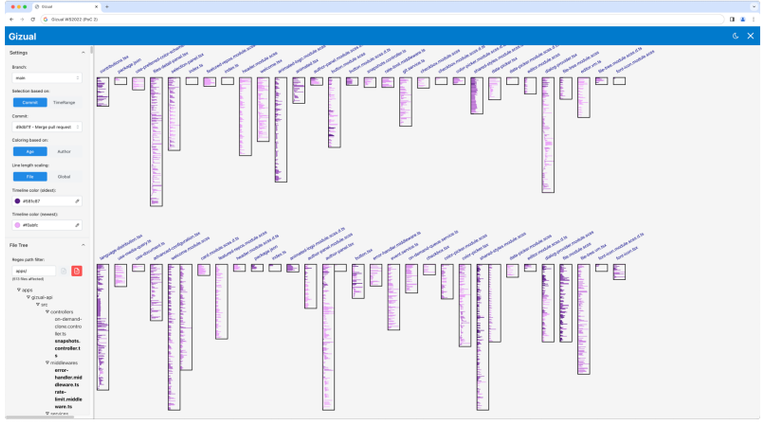
Gizual

Gizual Proofs of Concept [2022-2024]

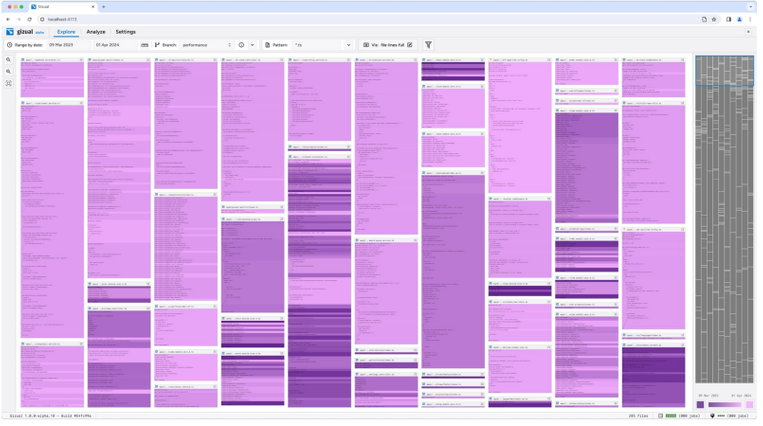
gizual.com



POC1[1] [May 2022 → June 2022]



POC2[2] [Dec 2022 → Jan 2023]



Alpha [Mar 2023 → Apr 2024]

1. *Gizual – Repository Visualisation for Git* [SS 2022]; seminar paper; https://gizual.com/resources/gizual-paper-ss2022.pdf
2. *Gizual – Repository Visualisation for Git* [WS 2023]; seminar paper; https://gizual.com/resources/gizual-paper-ws2022.pdf

Gizual [2022→]

gizual.com

- Intuitive Insights:
  Line-oriented visualisation with age and author colour-coding.
- High Performance:
  Fast visualisations on demand.
- Privacy First:
  Local data never uploaded.
- Stack:
  TypeScript, React, Mantine, Rust, WebAssembly (libgit2), Web Workers.

Gizual, 2025.

Showcase / Demo

Gizual Scalability

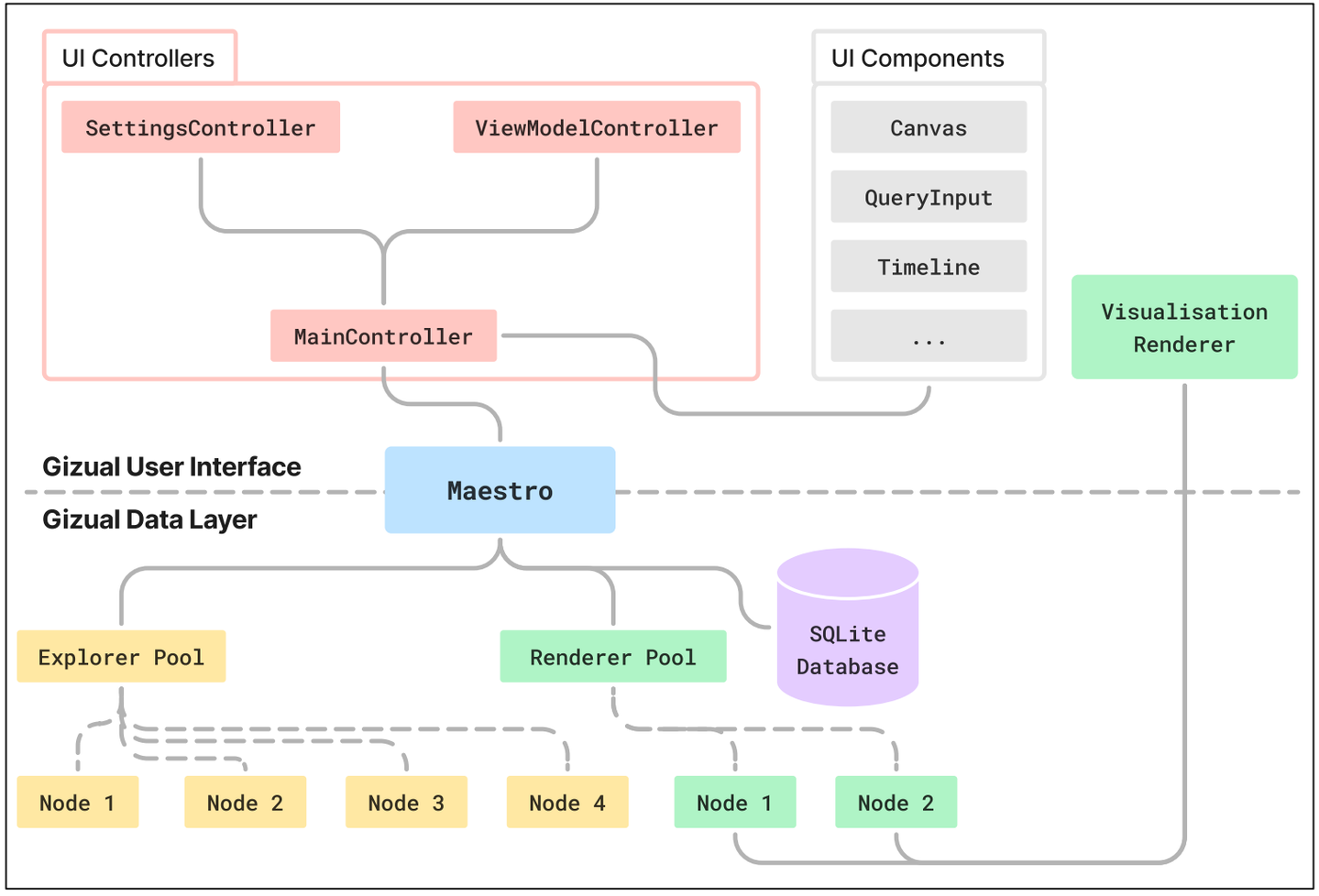
Repository	Size	.git/ Size	First Commit	# Commits	🖥️ Desktop	📱 Mobile
Vue 2	small	35 MB	10 Apr 2016	6,694	✅	✅
React	medium	657 MB	29 May 2013	25,873	✅	🚧
VS Code	large	1,025 MB	13 Nov 2015	134,419	✅	🚧
Linux Kernel	very large	5,836 MB	16 Apr 2005	1,310,317	❓	❓

The four GitHub repositories used to test Gizual. Statistics collected on 29 Oct 2024

Gizual Architecture

- User Interface
  by Andreas Steinkellner
  - Visualisation.
  - User Interaction.

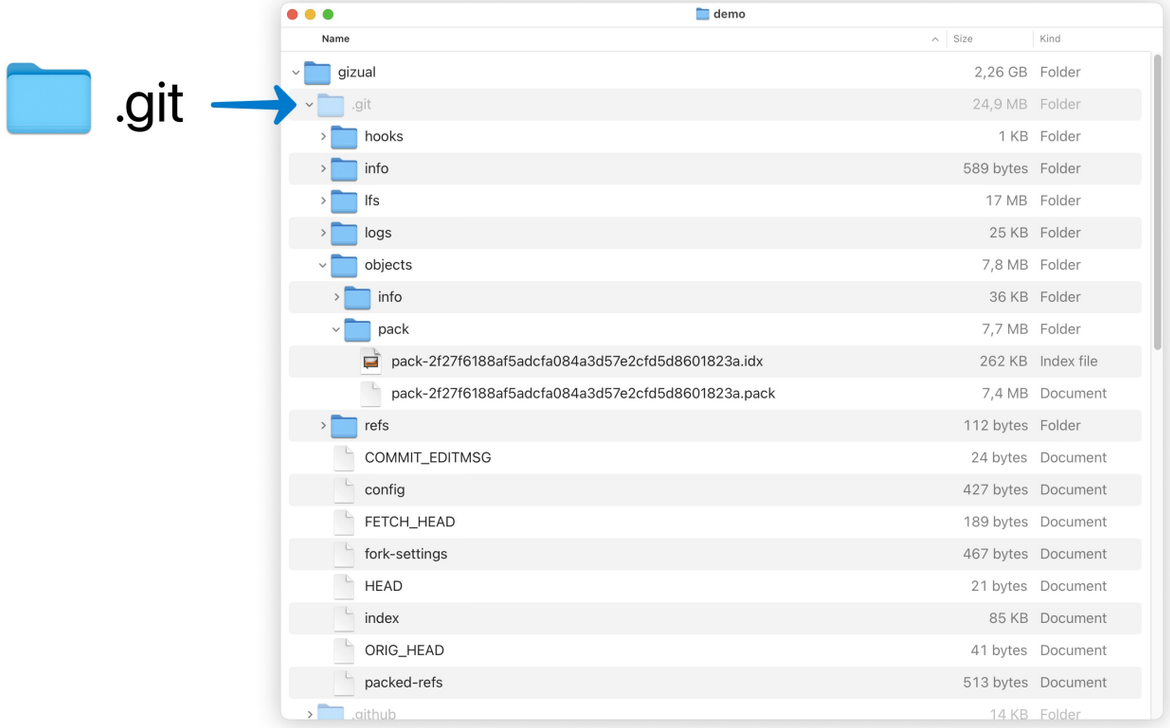
- Data Layer
  by Stefan Schintler
  - In-browser data processing (WebAssembly).
  - Broad browser support.
  - Performance (Pools of WebWorkers).



Gizual's Architecture

Gizual Data Layer

Loading Data into the Browser



File I/O

- File System Access API[1]
  - Supported by Chromium.
- Drag and Drop API[2]
- Origin Private File System[3]
  - Similarities to File System Access API.
  - Broad Browser Support.
  - Not stored in memory.

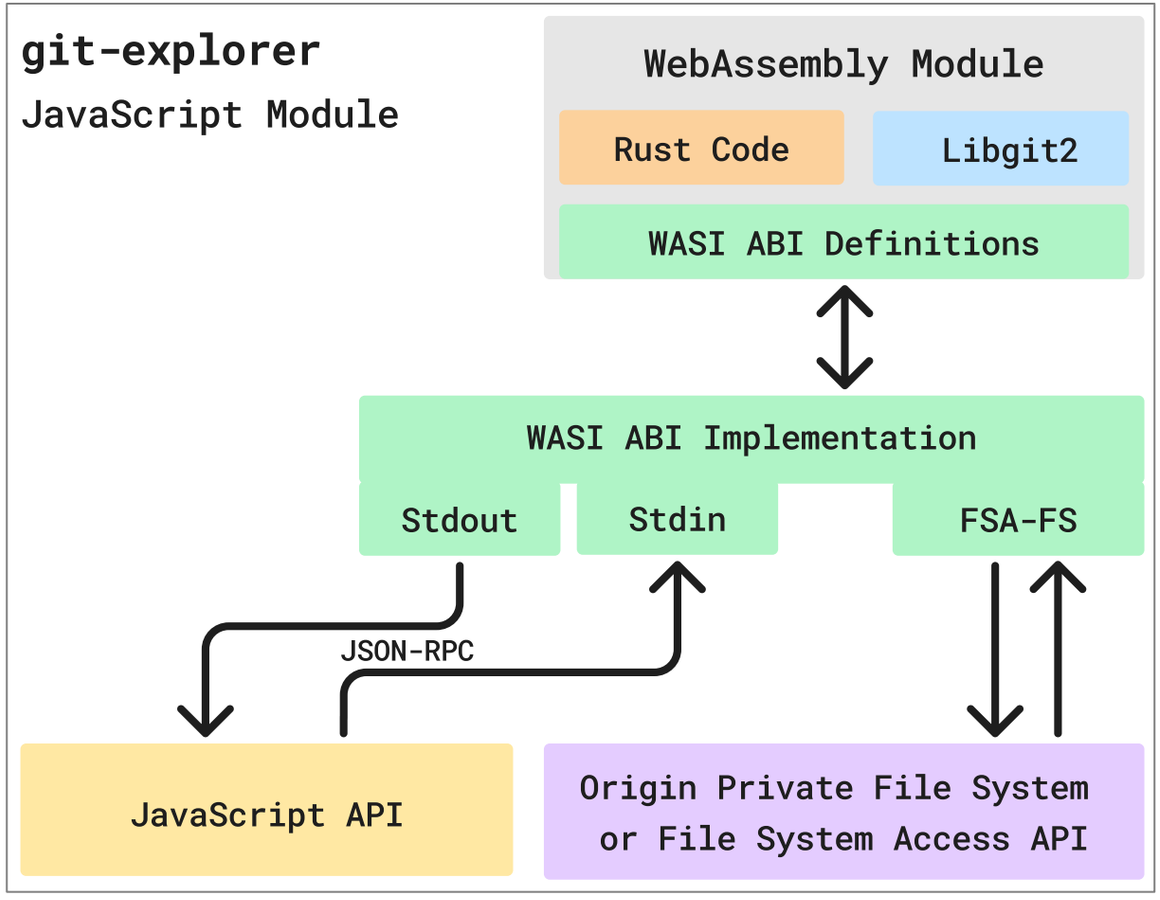
	Input	Storage
Chromium Chrome, Edge, Opera	File System Access API	
Firefox Waterfox, SeaMonkey	Drag & Drop API	Origin Private File System
Safari	Drag & Drop API	WIP[3] 🚧
Safari on iOS	only Remote-Clone	Origin Private File System
Chrome on Android	only Remote-Clone	Origin Private File System

Browser support with File I/O.

1. File System Access API; https://wicg.github.io/file-system-access/
2. Drag and Drop API; https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API
3. Origin Private File System; https://developer.mozilla.org/en-US/docs/Web/API/File_System_API/Origin_private_file_system
4. Safari – Incomplete OPFS Implementation; https://bugs.webkit.org/show_bug.cgi?id=254726

Native Code in the Browser

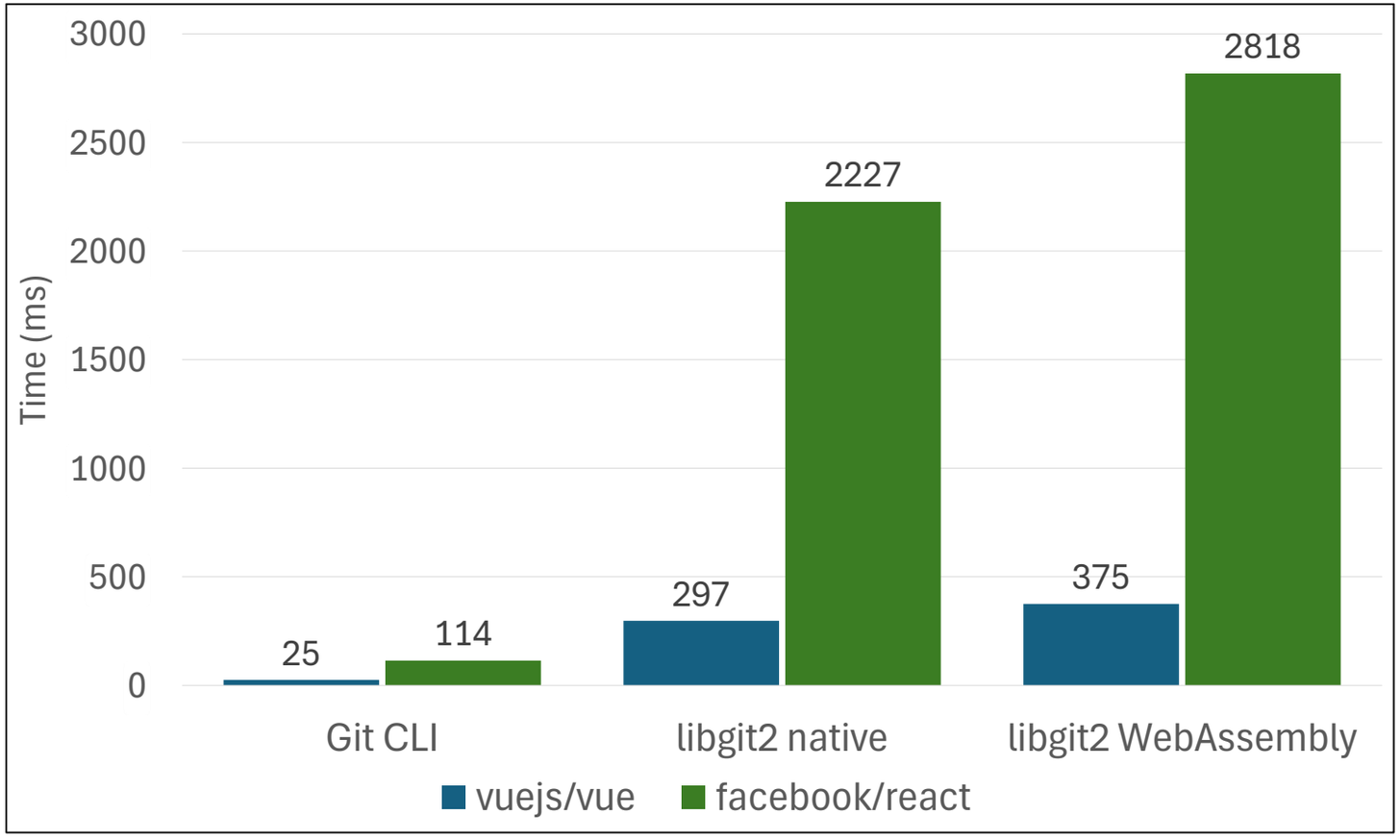
- Libgit2 C library[1]
- WebAssembly[2]
- Web Assembly System Interface[3]



Overview of git-explorer module.

1. Libgit2 Library; https://libgit2.org/
2. WebAssembly; https://developer.mozilla.org/en-US/docs/WebAssembly
3. WebAssembly System Interface (WASI); https://wasi.dev/

Blame Performance

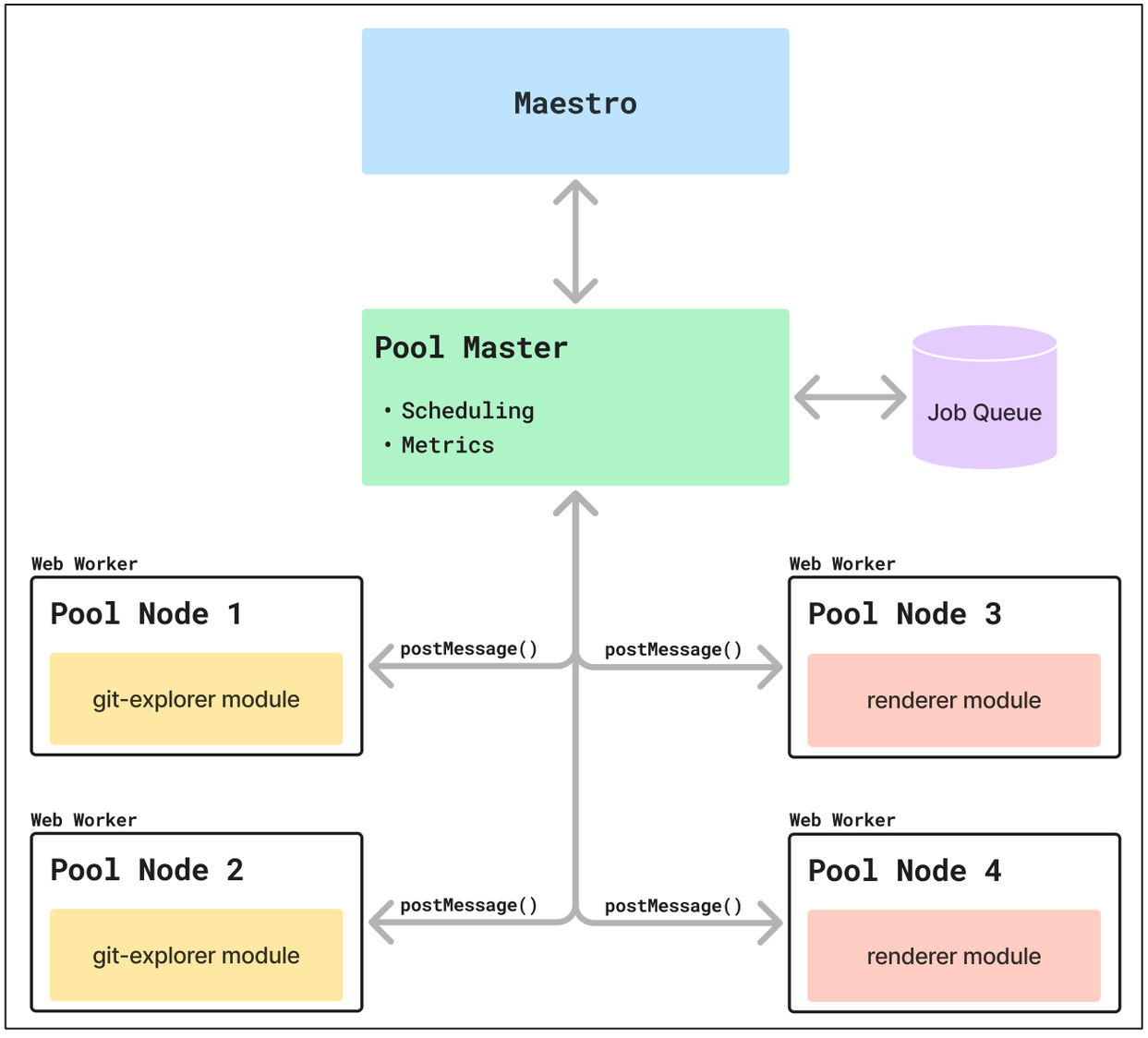


Comparison of blame performance; Blame was calculated for the single file package.json in both repositories on MacBook Pro 2023, M3 Max, 48 GB RAM.

Multi-Threading

- Pools of Web Workers for:
  - Data processing.
  - Rendering.
- Prioritised job queue.
- Interval-based scheduling.

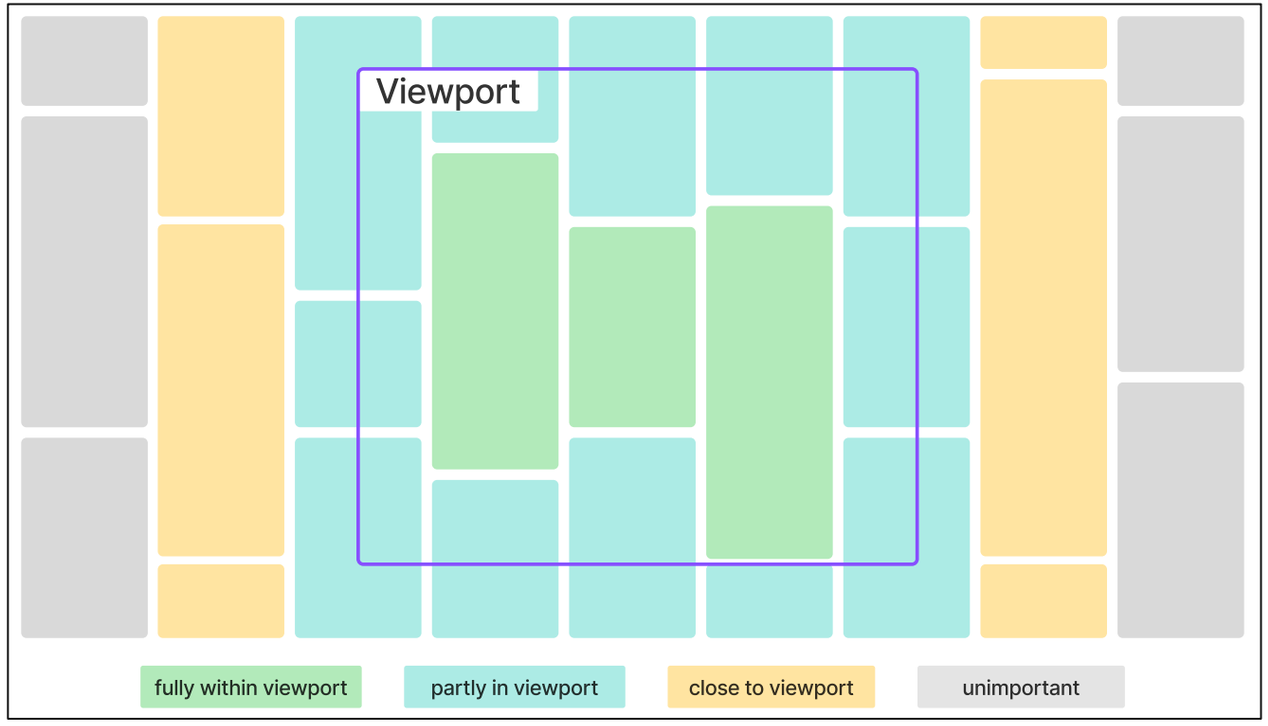
Worker usage: Git exploration (left) vs. rendering (right).



Pools of Web Workers.

Multi-Threaded Rendering

- Intersection Observer API[1].
- Priorisation based on viewport.
- Increase perceived performance.

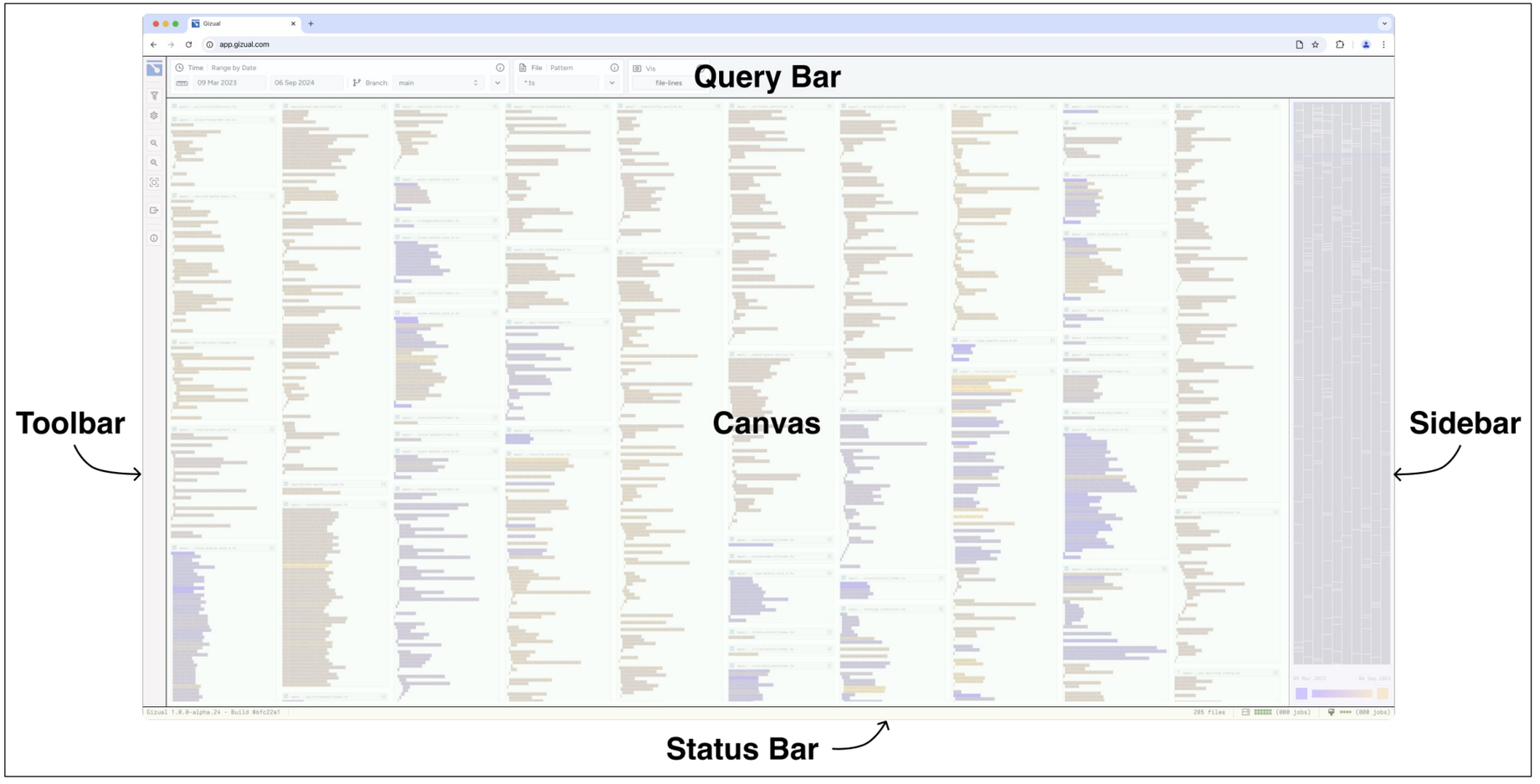


Priority of tiles on Gizual's Canvas.

1. Intersection Observer API; https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API

Gizual User Interface

User Interface Regions



Gizual's main user interface with annotated regions.

Rendering of Visualisation Canvas

📋 Requirements:

- Parallelised rendering.
- Distributed workload across web workers.
- Reusable for different visualisations and outputs.

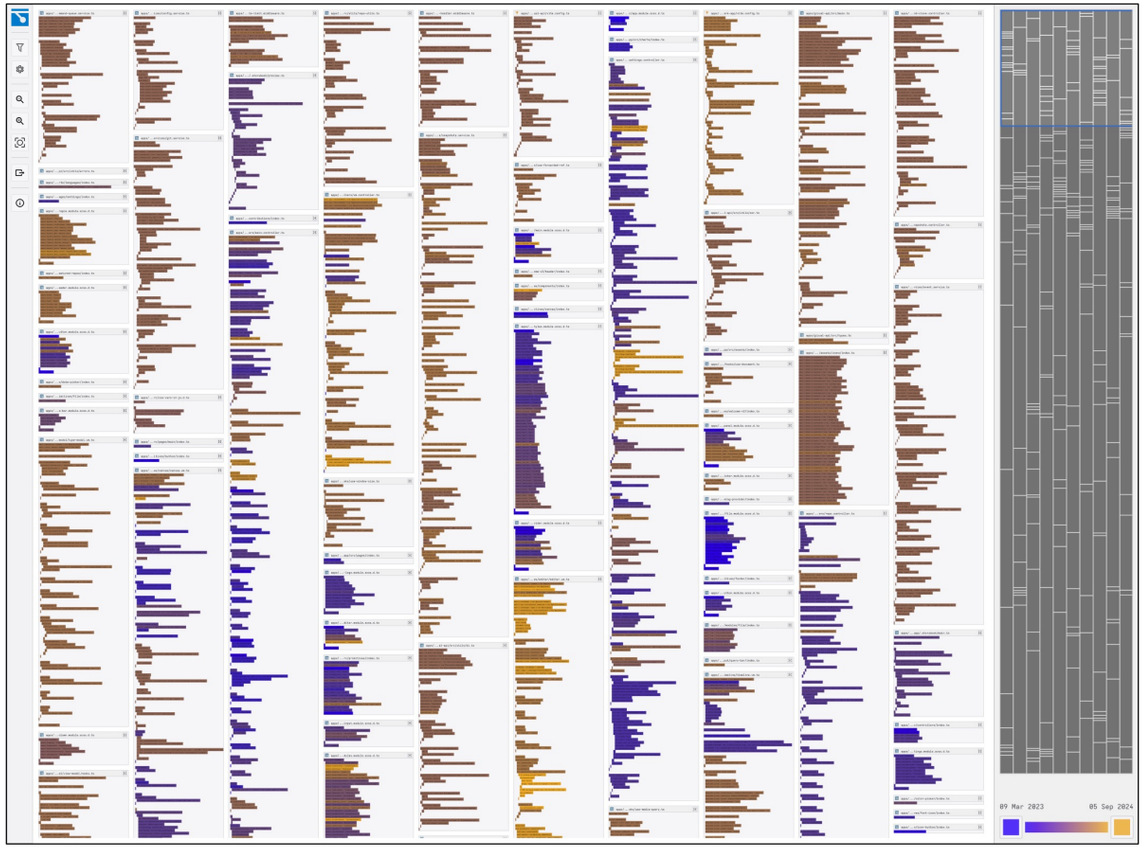
💡 Approach:

- Rendering tasks for each tile (file).
- Generalised rendering interface.
- Class-based file and module structure.

Canvas (1)

📋 Requirements:

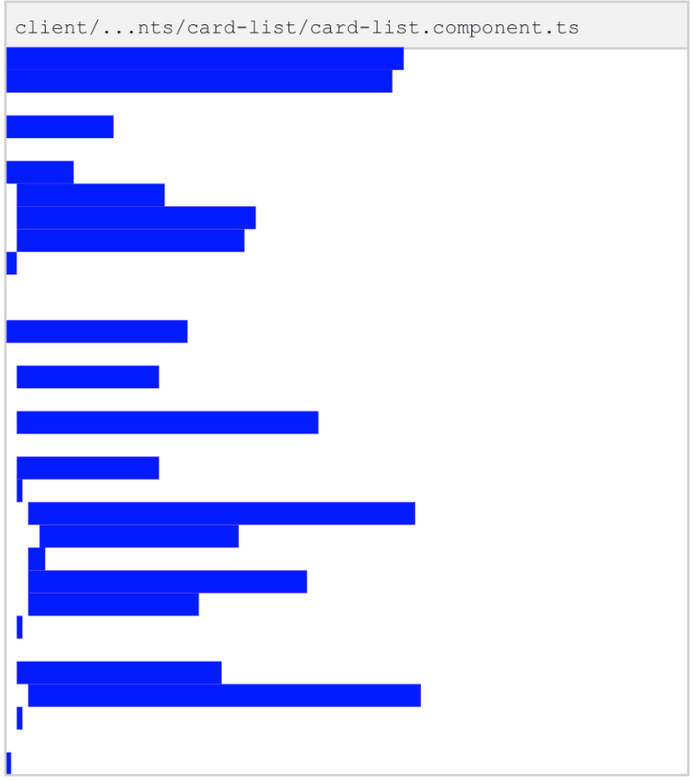
- Interactive, with high performance.
- Display hundreds (thousands) of files.
- Stable layout.
- Easily navigable.
- Good user experience on various viewport sizes and input devices.



Gizual's Canvas component.

Canvas (2) – Tile Structure

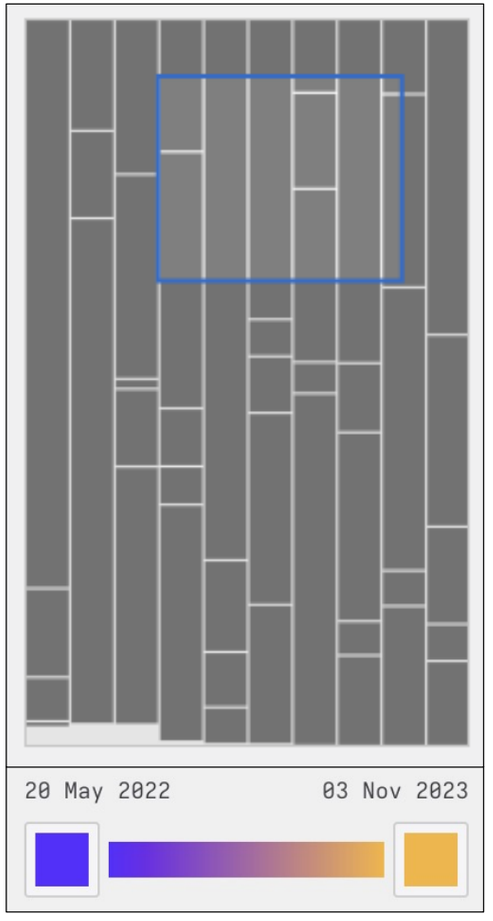
1. SVG Frame
   - `<foreignElement>` for interactive elements.
   - File name as text (Fonts: `Courier New` for export, otherwise `Iosevka`).
2. Pre-rendered image.
3. Future work: Interactive Annotation Layer.



Export of a single file tile in Gizual.

Canvas (3)

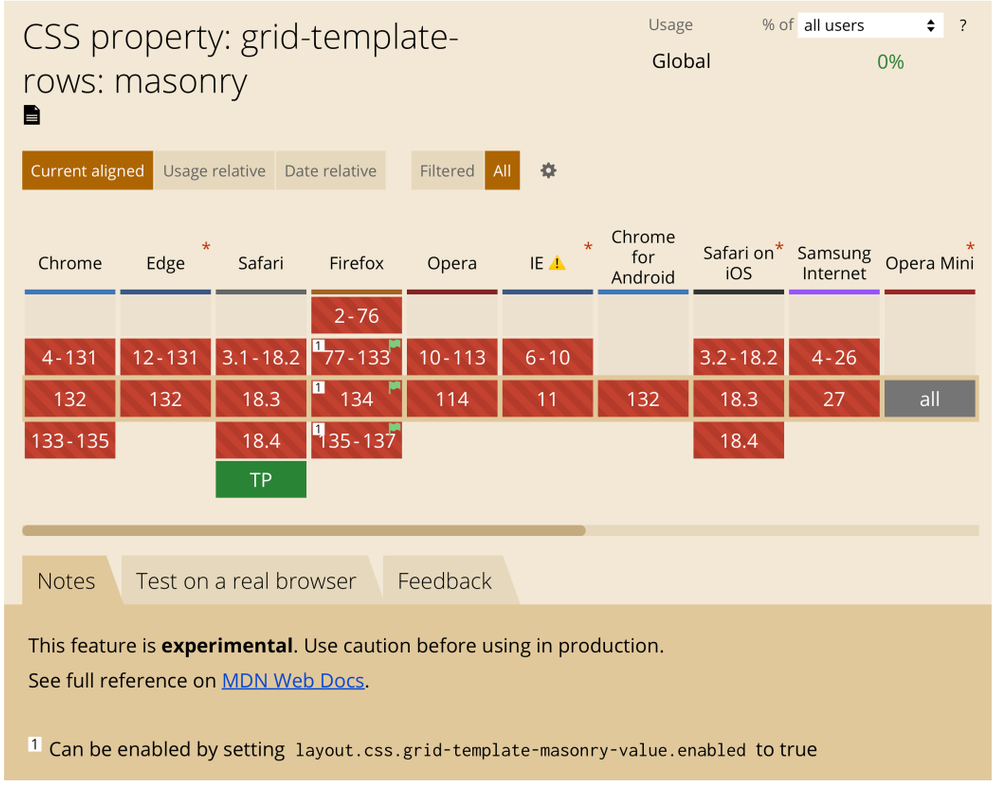
- Intuitive interactivity:
  - Zoom
  - Pan
  - Pinch
- Custom minimap implementation for overview in large visualisations.
- Custom legend component.
  - Quick access to gradient colours.
  - Show selected time range.



Minimap and legend.

Masonry Layout

- Why not CSS Grid?
  - `grid-template-rows: masonry`
  - Not available (still experimental)
  - Last W3C Working Draft: 03 Oct 2024[1]
- Custom JavaScript implementation with Flexbox columns as alternative.

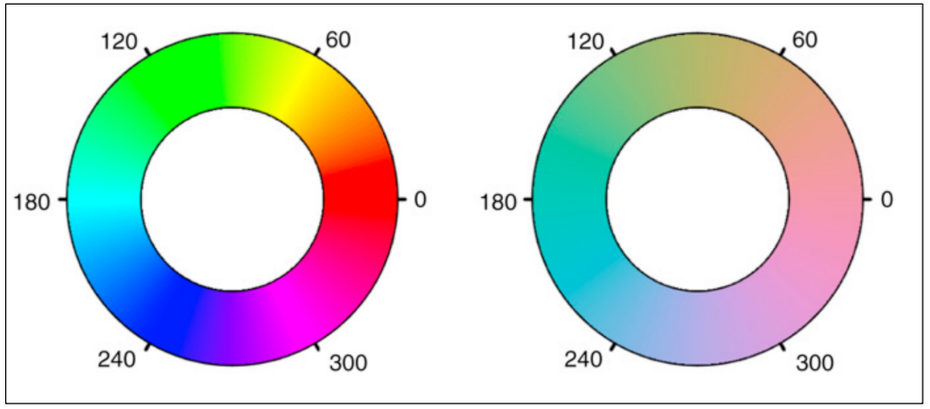


Browser support for `grid-template-rows: masonry`.
[Screenshot taken by the author of this presentation from caniuse.com[2]]

1. CSS Grid Layout Module Level 3; W3C Working Draft; 03 Oct 2024; https://www.w3.org/TR/css-grid-3/
2. Support for `grid-template-rows: masonry`; caniuse.com; 05 Feb 2025; https://caniuse.com/mdn-css_properties_grid-template-rows_masonry

Visual Encoding (1)

- Colour lines of code based on age or author.
- Age encoding: Gradient with two user-defined endpoints.
- Author encoding: HCL palette with custom user overrides.
  - RGB colour space not uniformly distributed → use HCL colour space.
  - Custom algorithm based on d3-color[1] for stable results.



Comparison of HSV-based (left) and HCL-based (right) colour wheel.
[Image extracted from Zeileis et al.[2]]

1. d3-color: https://d3js.org/d3-color
2. *Escaping RGBland: Selecting Colors for Statistical Graphics;* Zeileis, Achim and Hornik, Kurt and Murrel, Paul [2009]; Computational Statistics & Data Analysis 53.9 (Jul 2009). 10.1016/j.csda.2008.11.033

Visual Encoding (2)

Model	Parameters	Details
RGB	Red, Green, Blue	Non-uniform, device dependent.
HSL	Hue, Saturation, Lightness	Perceptually-oriented, based on RGB.
HSV	Hue, Saturation, Value	Used interchangeably with HSL.
CIE XYZ	Tristimulus response (eye)	Perceptually uniform reference model, based on human vision experiments.
CIE Lab	Lightness, Green-to-red, Blue-to-yellow	Perceptually uniform, device independent.
HCL	Hue, Chroma, Luminance	Perceptually uniform, based on CIE.
OKLCH	Luminance, Chroma, Hue	Perceptually uniform, optimised for digital screens.

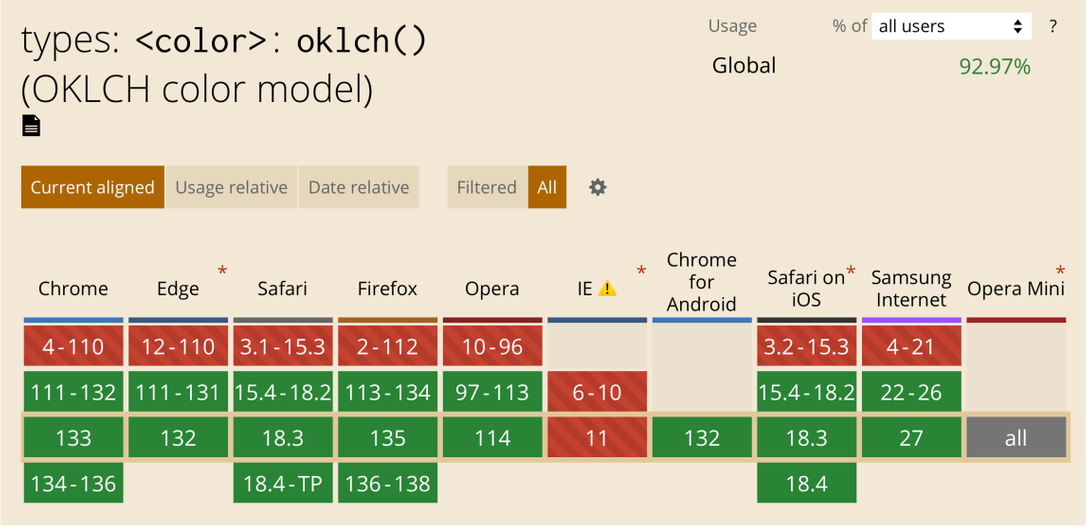
Comparison of commonly used colour spaces.
[Table created by the author of this presentation, based on data from Noor et. al.[1] and the W3C Standard for CSS Color Module Level 4[2].

HCL colour band, created in Gizual for a set of ten authors.

1. *Understanding Color Models: A Review*; Ibraheem, Noor & Hasan, Mokthar & Khan, Rafiqul Zaman & Misha, Pramod [2012]; ARPN Journal of Science and Technology. https://www.researchgate.net/publication/266462481_Understanding_Color_Models_A_Review
2. CSS Color Module Level 4; W3C Candidate Recommendation Draft; 13 Feb 2024; https://www.w3.org/TR/css-color-4/#ok-lab

Visual Encoding (3)

- Now: Would use CSS OKLCH[1] colour space.
- Contrast values of colours more evenly distributed.
- Good support in all major browsers.
- Perceptual uniformity without JavaScript.
- For design tools: Use plugins / external colour pickers.

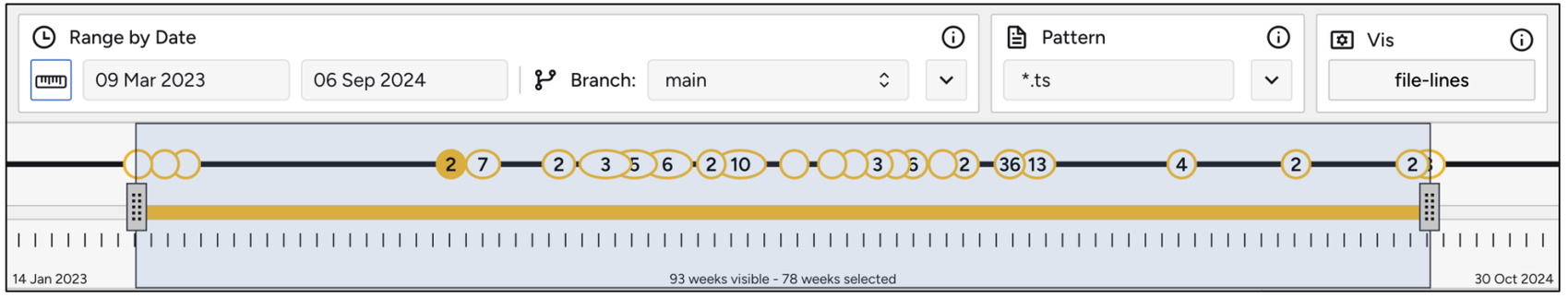


Browser support for `<color>: oklch()`
[Screenshot taken by the author of this presentation from caniuse.com[2]]

1. OKLCH Color Picker & Converter; https://oklch.com/
2. Support for `<color>: oklch()`; caniuse.com; 10 Feb 2025; https://caniuse.com/mdn-css_types_color_oklch

Query Bar

- Modular setup, easily extendible.
- Consistent look and feel.
- Timeline as interactive SVG element.



Gizual's Query Bar, with the Timeline component expanded.

Thank You!

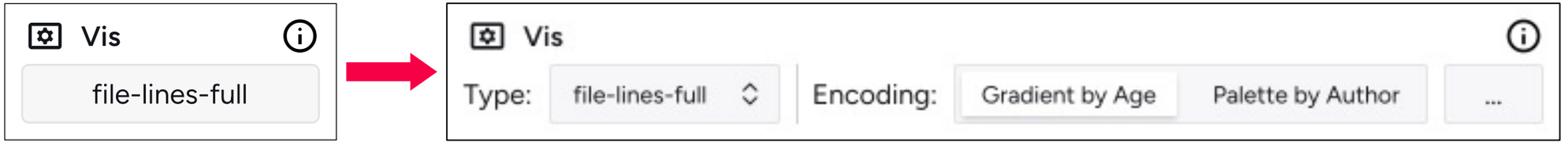
gizual.com

Future Work (1)

- Canvas performance improvements through SVG-based rendering.
- Custom analytics with charts.
- Timeline performance improvements through CSS transforms.
- Binning for visual encoding.
- Brushing for specific lines of code.
- Hierarchical structure overview (treemap).
- SVG export with file content.
- Native build (Electron/Tauri).
- New layouts for tiles (e.g. preserve alphabetic ordering by file name).
- Fulltext search

Future Work (2)

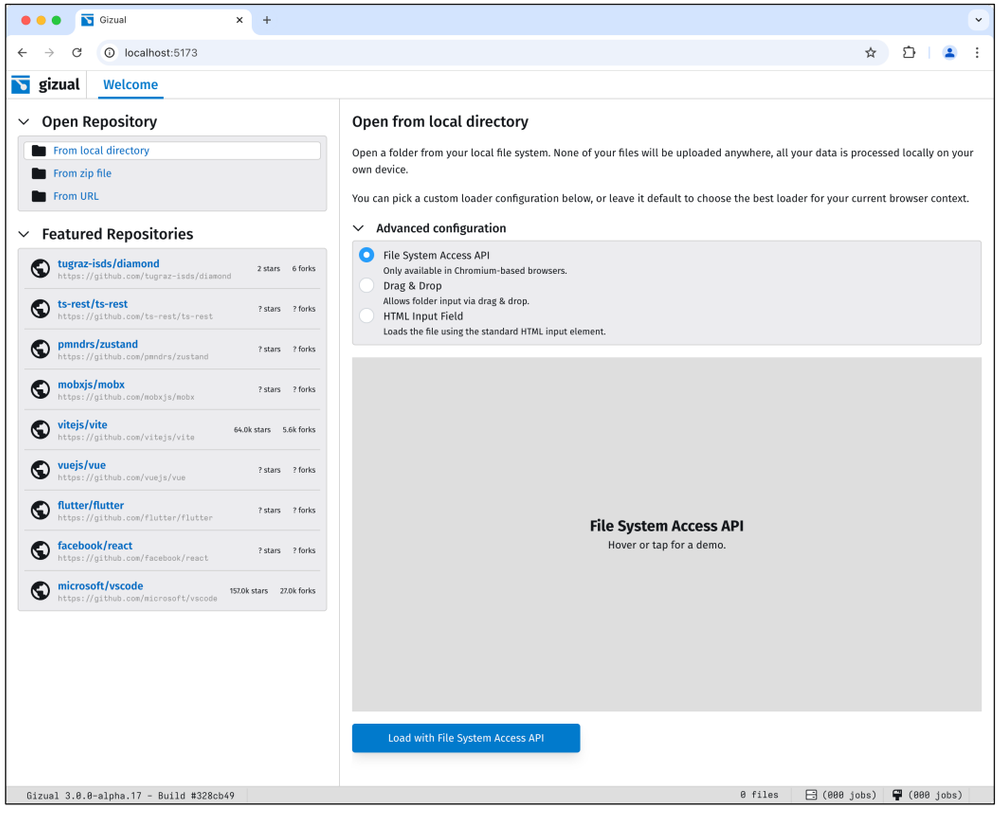
- Quick toggles: Visual encoding.



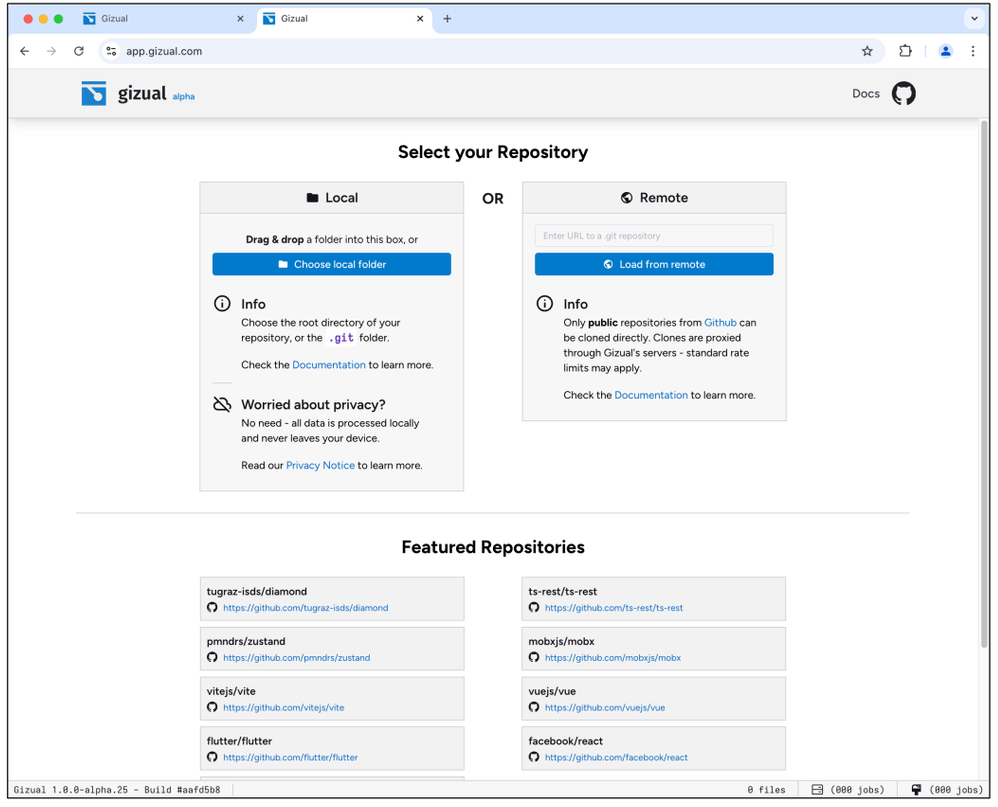
Ideas:

- Text diff between revisions.
- Merge conflict detector.
- Filter by commit message.

Welcome Screen: Iterative UX Improvements



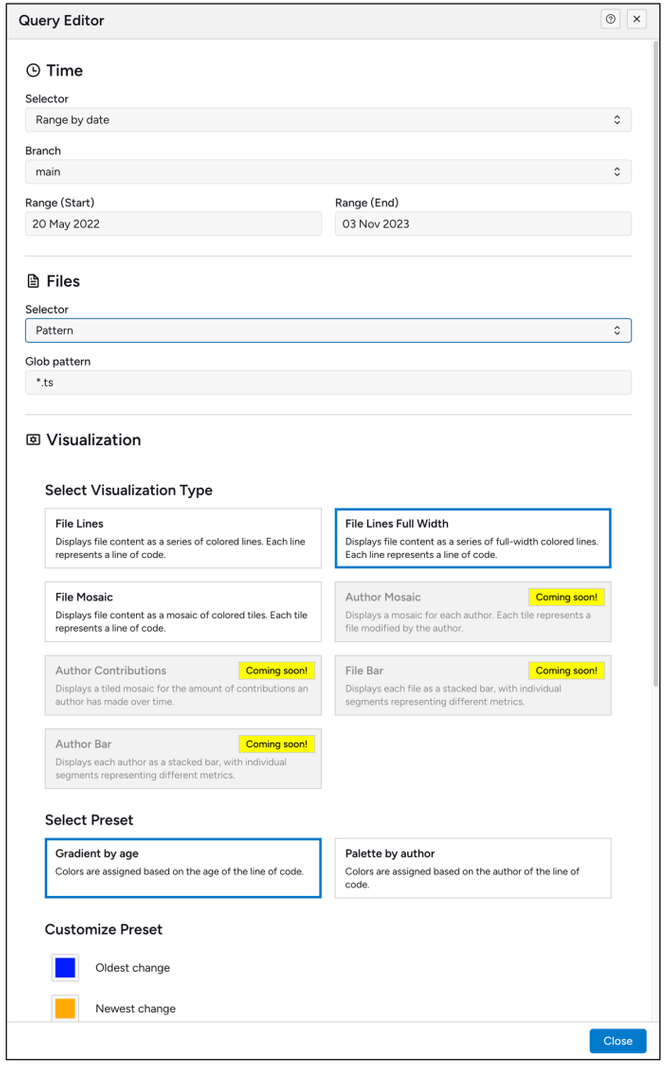
Gizual's welcome screen [Mar 2024].



Gizual's welcome screen [current].

Query Editor

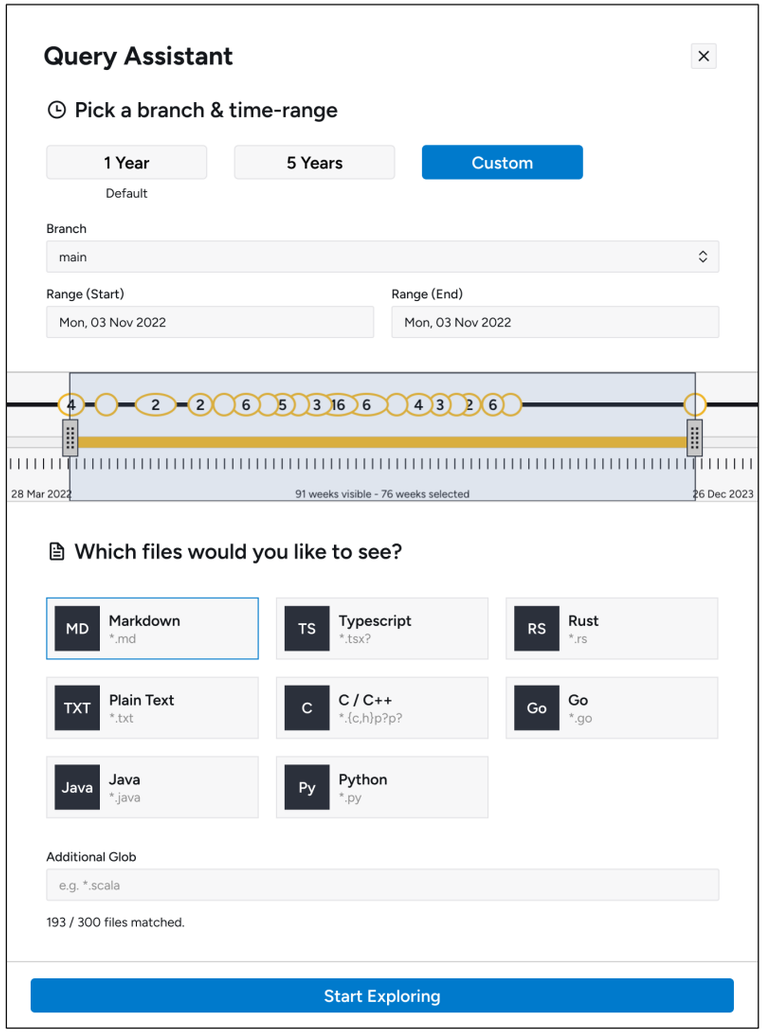
- Replaces Query Bar on devices with narrow viewports.
- Provides a simplified selection of modules.



Gizual's Query Editor.

Query Assistant (Draft)

- Easier way for new users to get started.
- Contains a subset of available options.
- Highly focused on selecting a range and files to start immediately.

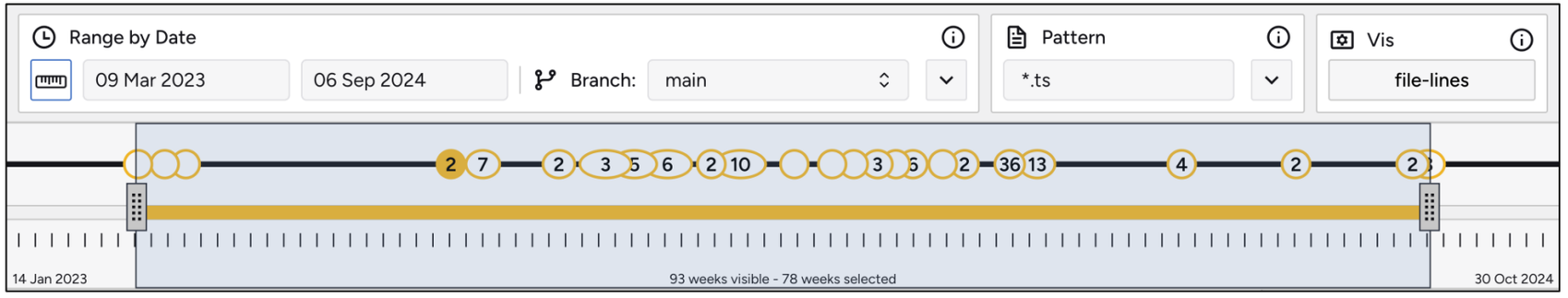


Gizual's Query Assistant.

Query Bar (1)

📋 Requirements:

- Everything accessible within few clicks.
- Modular setup, easily extendible.
- Consistent look and feel.

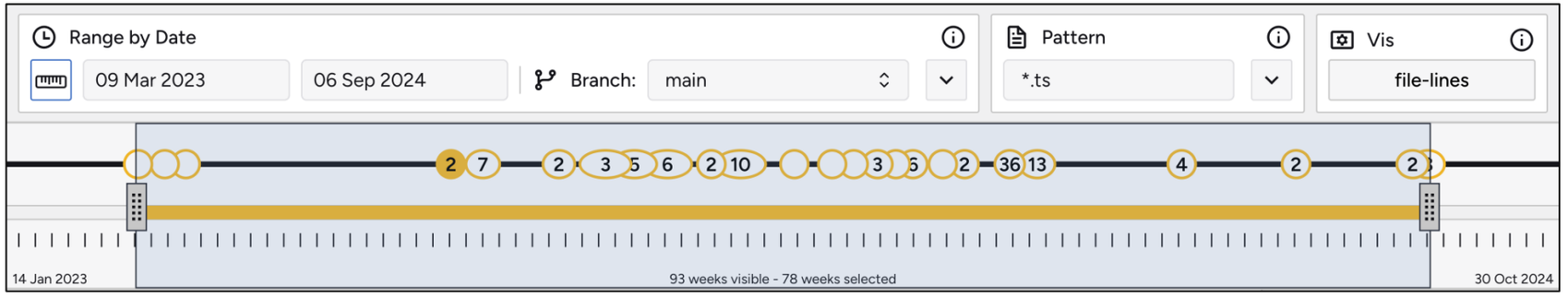


Gizual's Query Bar, with the Timeline component expanded.

Query Bar (2)

💡 Approach:

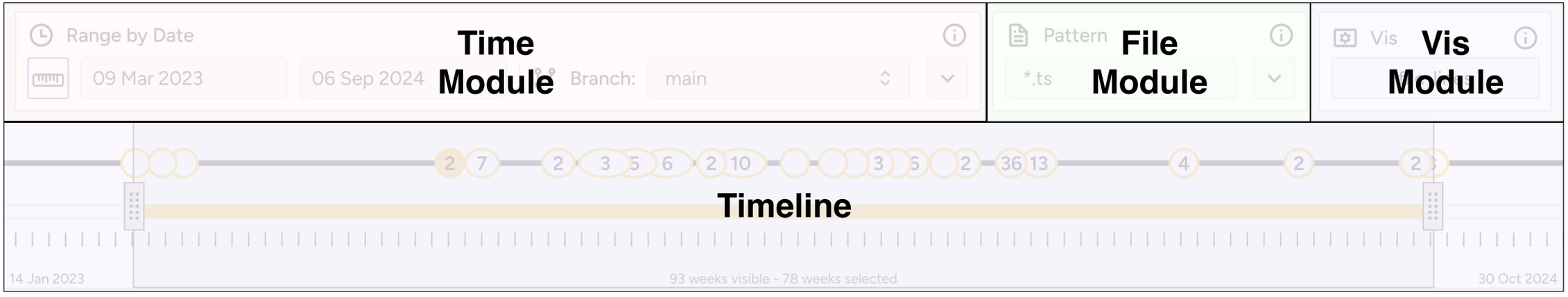
- Generalised base module exposes common functionality.
- Individual modules grouped by category.
- Modules directly attached to query interface via Maestro.



Gizual's Query Bar, with the Timeline component expanded.

Query Bar Modules

- Module groups provide interchangeable modules.
- Visual layout always consistent.
- Modules can be swapped with one button.

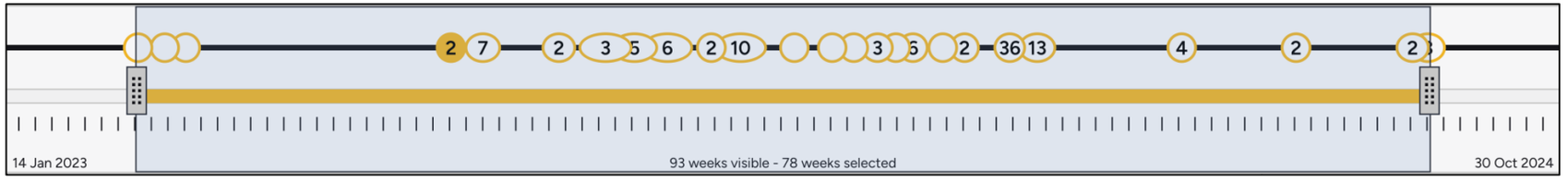


Gizual's Query Bar, annotated with module information.

Query Bar: Timeline (1)

📋 Requirements:

- Easy selection of time range.
- Visual display of commits.
- High performance.

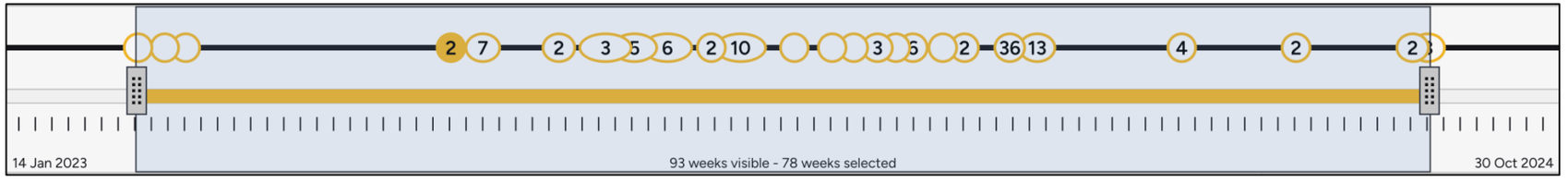


Gizual's Timeline component.

Query Bar: Timeline (2)

💡 Approach:

- SVG-based rendering with interactive HTML on top.
- Interactivity for zooming, panning and dragging (mouse and touch).
- Pre-rendering on left and right to improve performance.



Gizual's Timeline component.

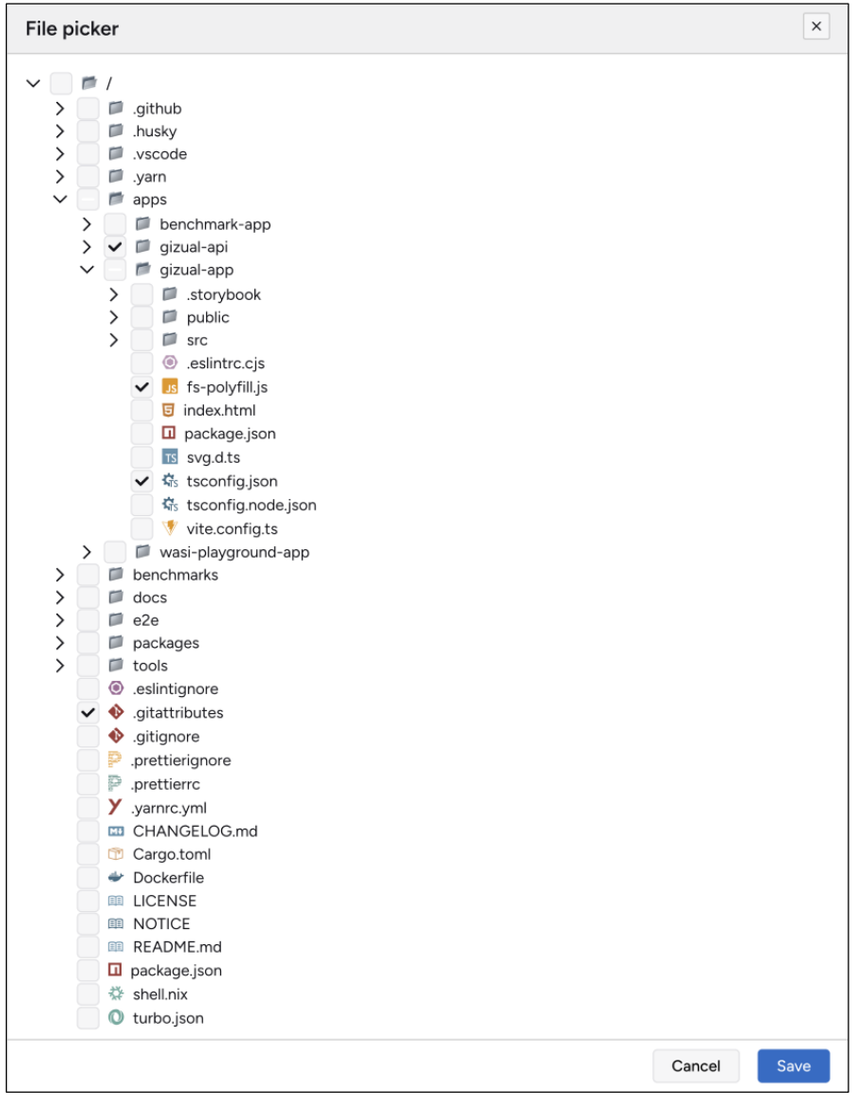
Query Bar: File Tree

📋 Requirements:

- Support for thousands of files.
- Partial selection for folders.
- File type icons.

💡 Approach:

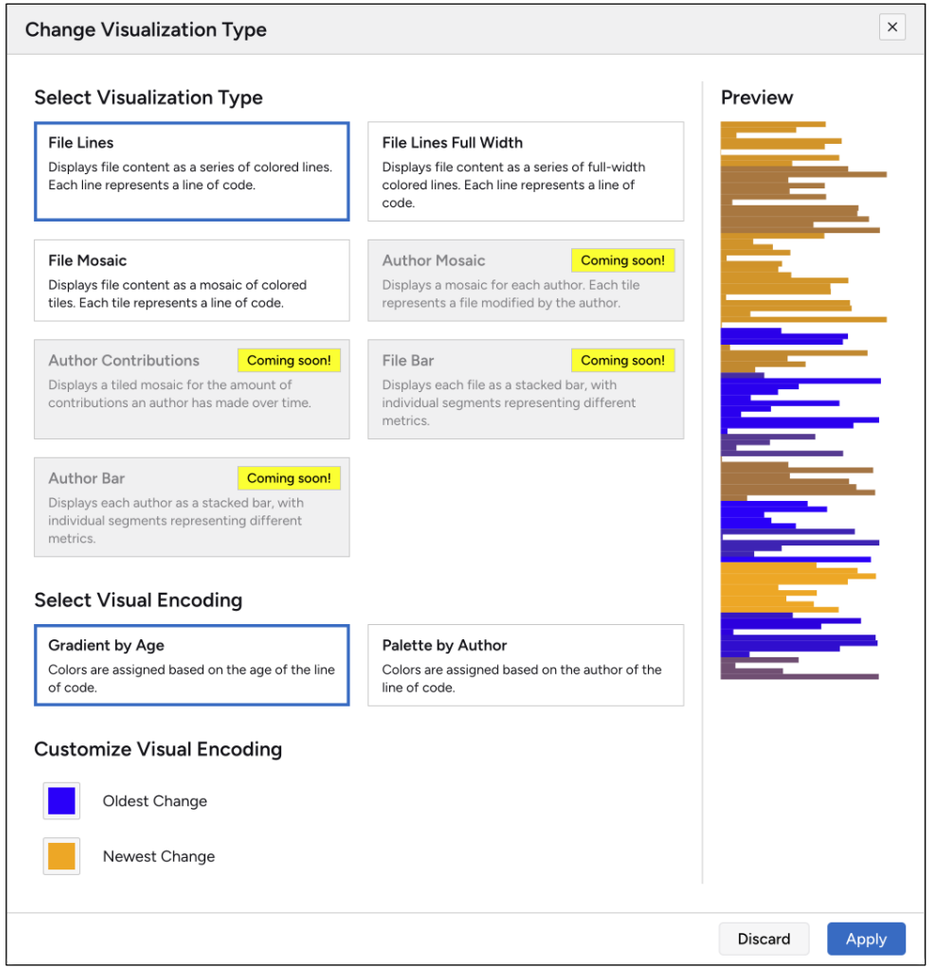
- Limited recursive pre-rendering (3 levels).
- Internal tree representation in flat structure.



Gizual's File Tree component.

Visualisation Type Dialogue

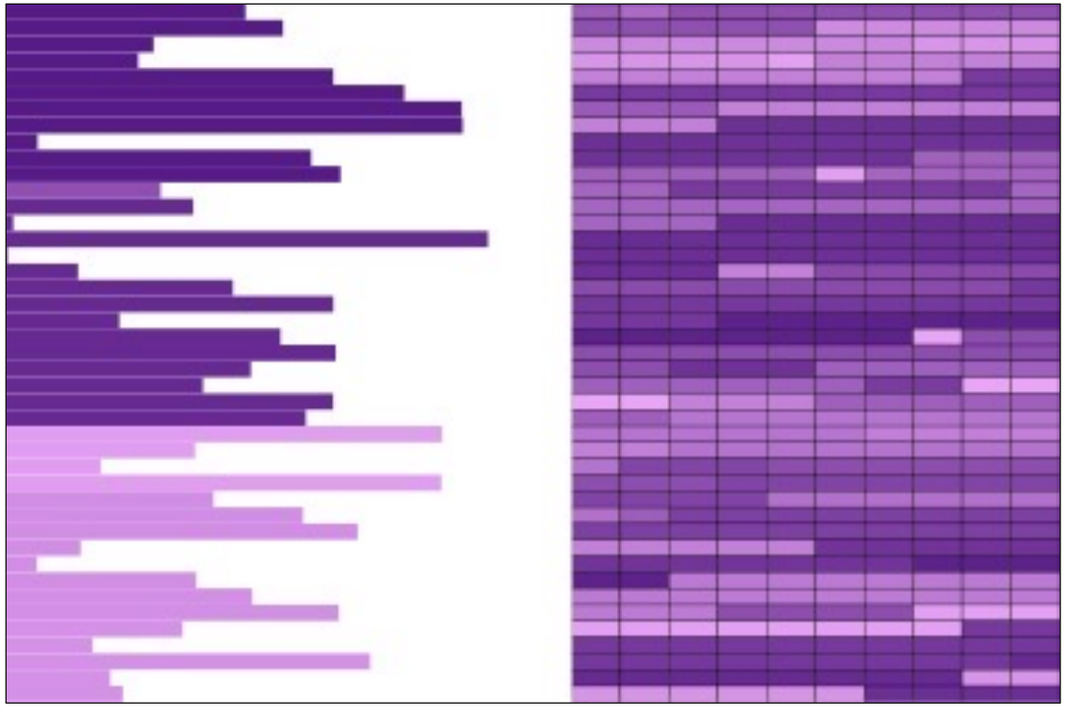
- Deliberate use of dialogue instead of direct display in Query Bar.
- Overview of visualisation types.
- Choice between visual encodings.
- Preview of selected visualisation type in selected encoding with customised colours.



Gizual's Visualisation Type Dialogue.

Mosaic Mode

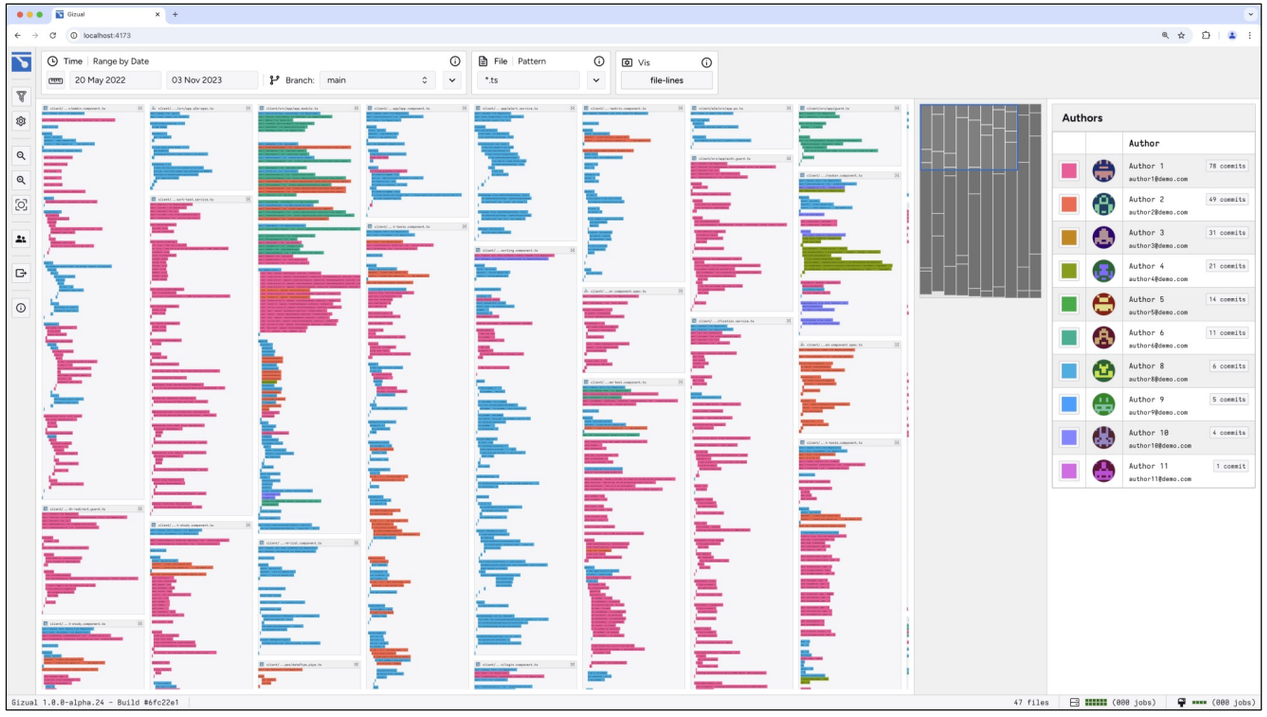
- Alternative visualisation mode.
- 10 lines of source code grouped into a single line of tiles.
- Condensed representation.
- No rendered text content.



Gizual's line mode (left) compared to mosaic mode (right).

Palette by Author Encoding

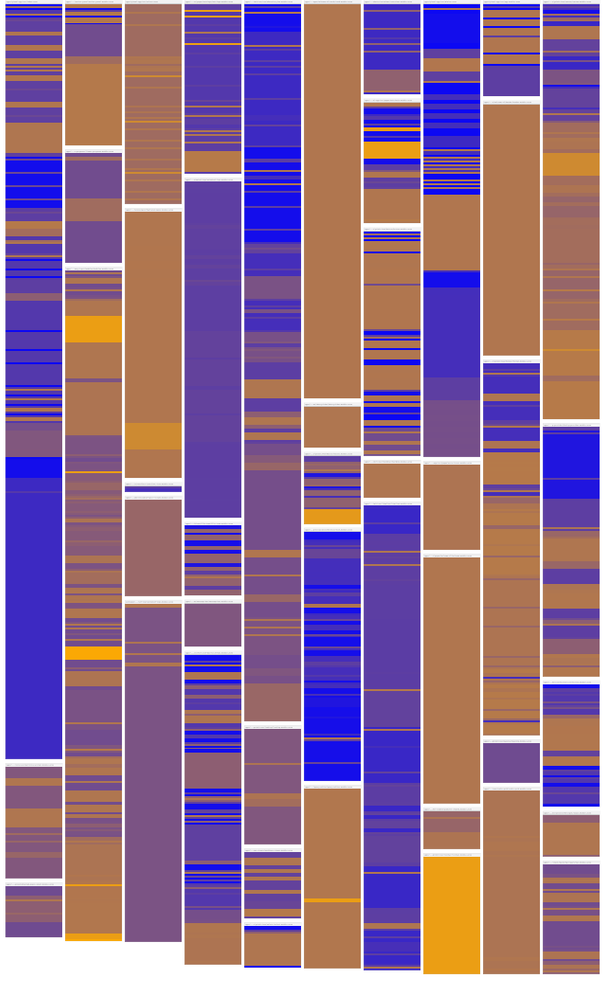
- Colour based on author of line of code instead of timestamp.
- HCL colour space for equal distribution per default.
- Author panel automatically visible.
- Colours can be customised by user.



Gizual's main interface with a visualisation in Palette by Author visual encoding.

SVG Export

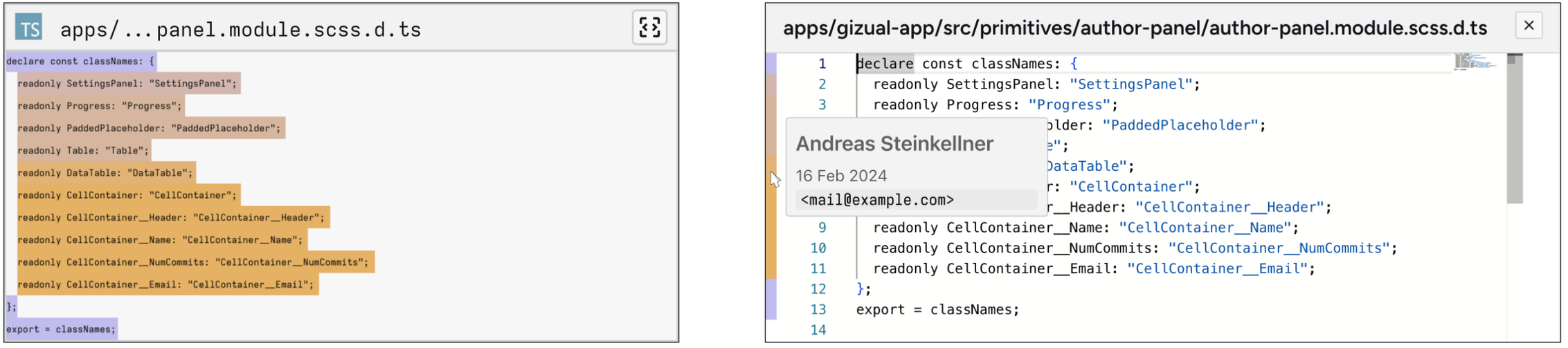
- Stable masonry algorithm.
- Files assigned into columns and sorted based on rendered height.
- File header rendered without interactive elements.



Gizual's SVG Export.

Integrated Code Editor

- Based on Monaco[1] code editor.
- Custom line gutter and tooltip.

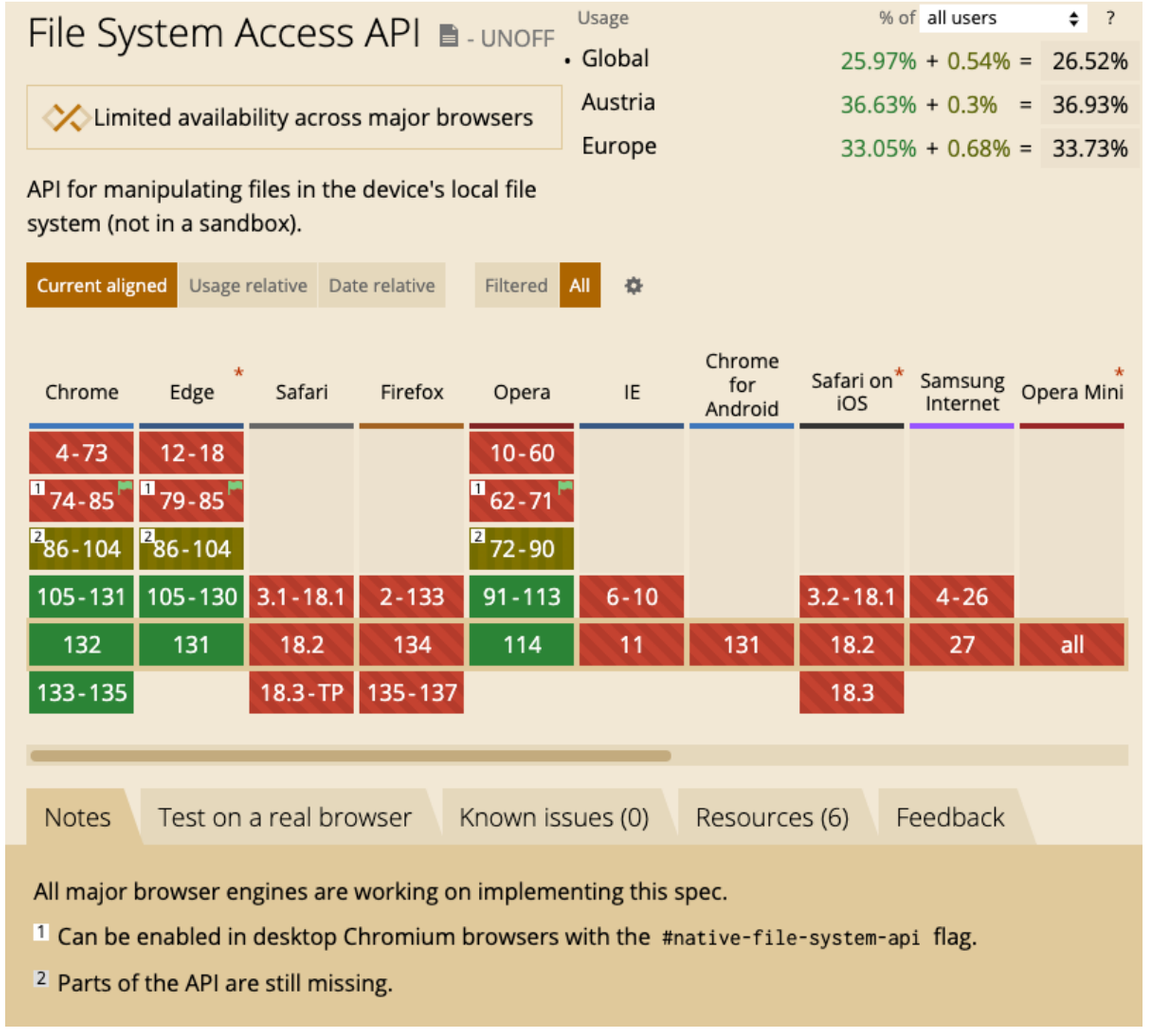


Gizual file content in visualisation (left) vs. file content in integrated editor (right).

1. Monaco; https://microsoft.github.io/monaco-editor/

File System Access API[1]

- Abstraction of file system.
- Mostly async API.
- Access local files directly.
- Supported by Chromium.



Browser support for the File System Access API.
[Screenshot taken by the author of this presentation from caniuse.com[2]]

1. File System Access API; https://wicg.github.io/file-system-access/
2. File System Access API Support; caniuse.com; 21 Jan 2025; https://caniuse.com/native-filesystem-api

Parsing Git Data

- Git Command Line[1]
  - Not portable (POSIX Shell etc.).
- Isomorphic-git Library[2] (JS)
  - No blame support.
- Gitoxide Library[3] (Rust)
  - No blame support (yet).
- Libgit2 Library[4] (C)
  - Mediocre blame performance[5].

1. Git Command Line; https://git-scm.com/
2. Isomorphic-git Library; Hilton, William; https://isomorphic-git.org/
3. Gitoxide Library; Thiel, Sebastian; https://github.com/Byron/gitoxide
4. Libgit2 Library; https://libgit2.org/
5. Libgit2 Performance Issue; https://github.com/libgit2/libgit2/issues/3027

WebAssembly[1]

- Modern binary instruction format.
- Widespread support, even within the browser sandbox.
- C / C++ / Rust / Go
- No Kernel ABI / Syscalls available.

Browser support for WebAssembly
[Screenshot taken by the author of this presentation from caniuse.com[2]]

1. WebAssembly; https://developer.mozilla.org/en-US/docs/WebAssembly
2. WebAssembly Support; caniuse.com; 21 Jan 2025; https://caniuse.com/wasmj

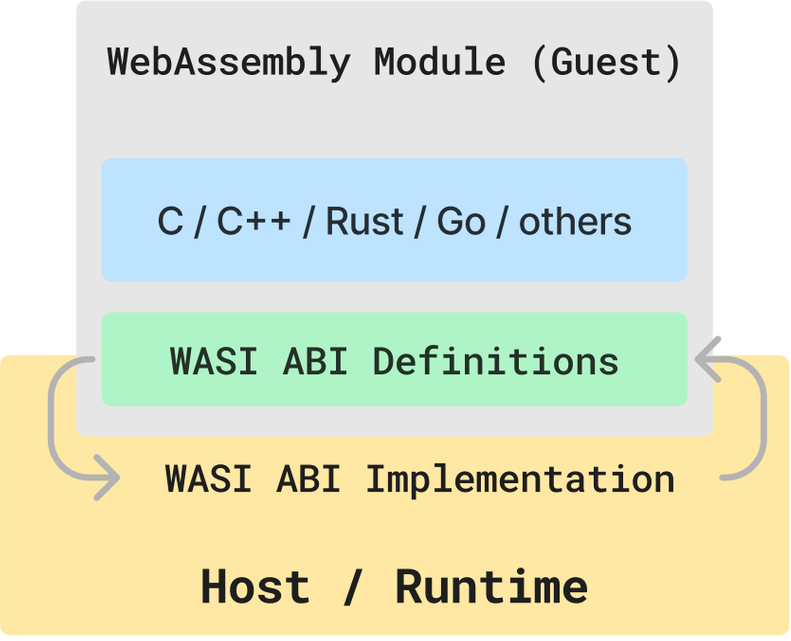
WebAssembly System Interface (WASI)[1]

- Used: preview1 (Version 0.1)
- Set of common API specifications.
- Command-Line Interface (CLI)-like behavior.

```ts
interface WasiFunctions {
  args_sizes_get(argc: number, argv_buf_size: number): number;
  args_get(argv: number, argv_buf: number): number;

  fd_close(fd: number): number;
  fd_read(fd: number, iovs_ptr: number, iovs_len: number, nread_ptr: number): number;
  fd_write(fd: number, iovs_ptr: number, iovs_len: number, nwritten_ptr: number): number;
  fd_filestat_get(fd: number, filestat_ptr: number): number;
  // ... and many more
}
```

TypeScript Interface showcasing API of WASI

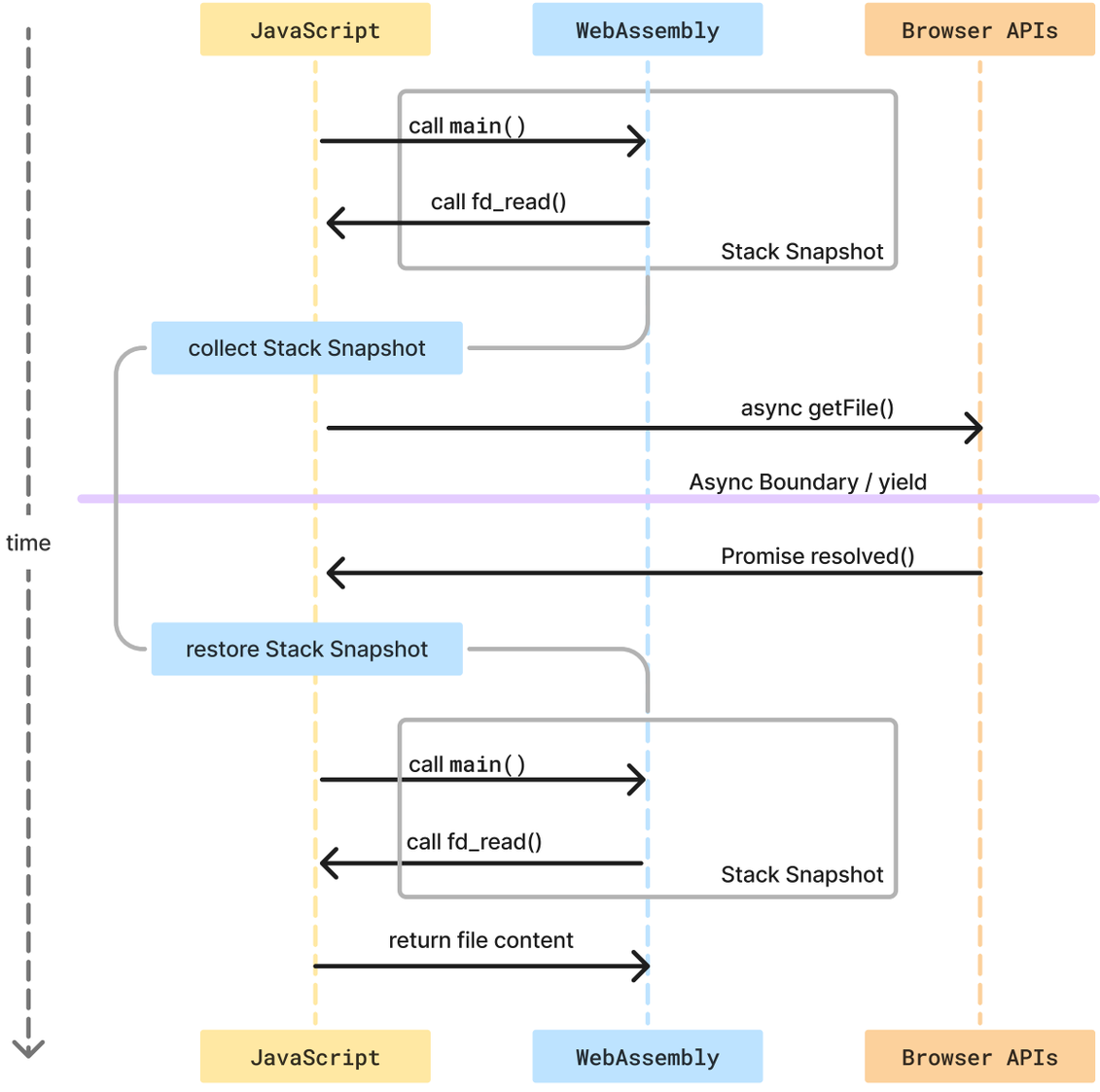


WASI Visualisation

1. WebAssembly System Interface (WASI); https://wasi.dev/

Async File I/O Interface

- Based on approach by A. Zakai[1].
- Using Asyncify[2] .
- Drawback:
  Significant increase of module size.



Simplified diagram showcasing Asyncify

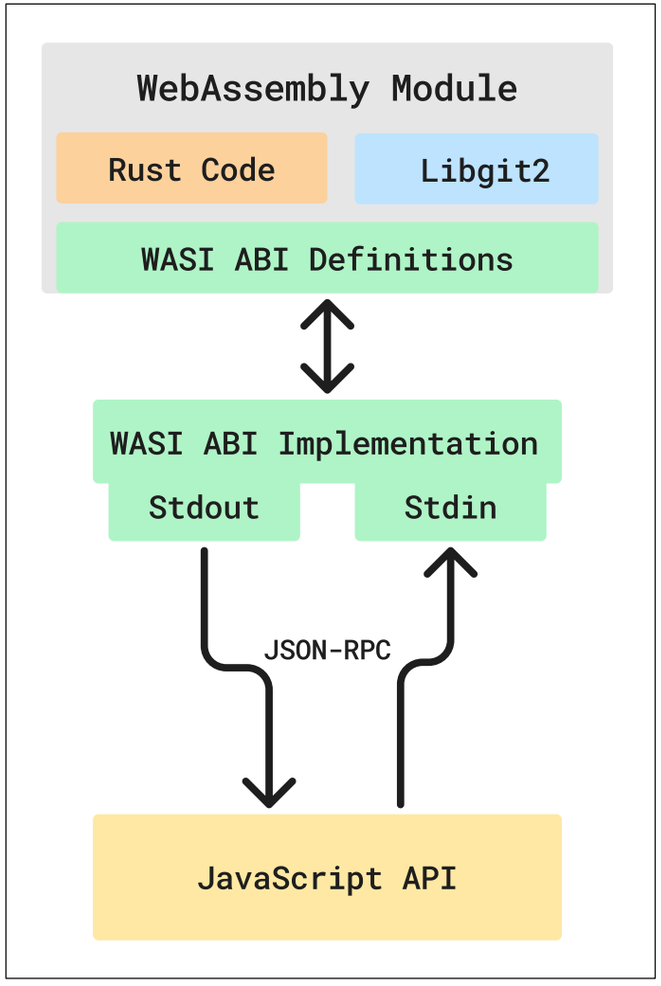
1. Zakai, Alon; https://kripken.github.io/blog/wasm/2019/07/16/asyncify.html
2. Binaryen Asyncify; https://github.com/WebAssembly/binaryen/blob/main/src/passes/Asyncify.cpp

JavaScript to Rust Command Interface

- JSON-RPC
- Use of FD stdin / stdout.
- Async wait for next job.

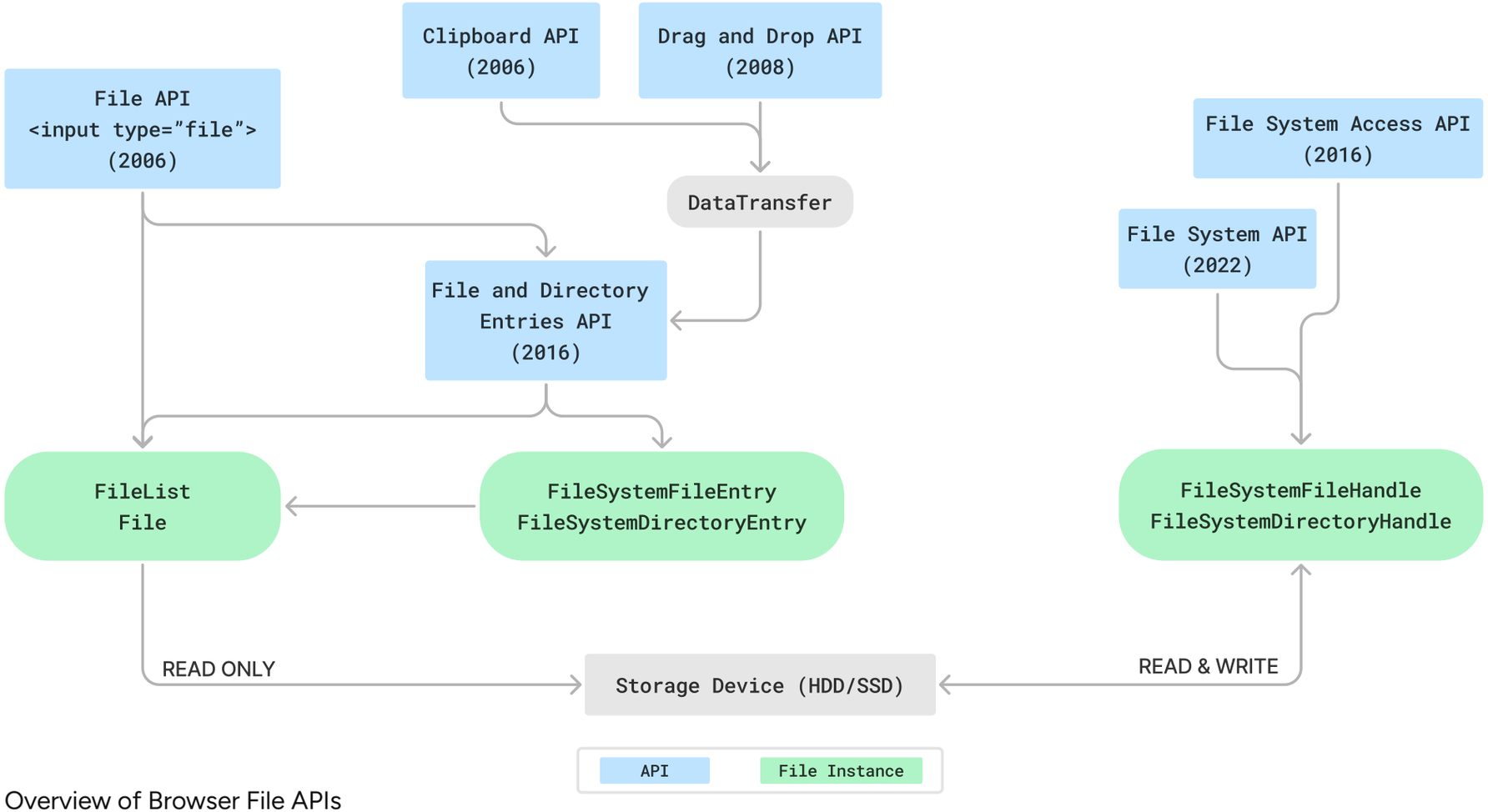
```
{
   jsonrpc: "2.0",
   id: 5,
   priority: 1,
   method: "get_blame",
   params: {
      path: "./package.json",
      rev: "v2",
      sinceRev: "v1",
   },
};
```

JSON-RPC Request used within Gizual.



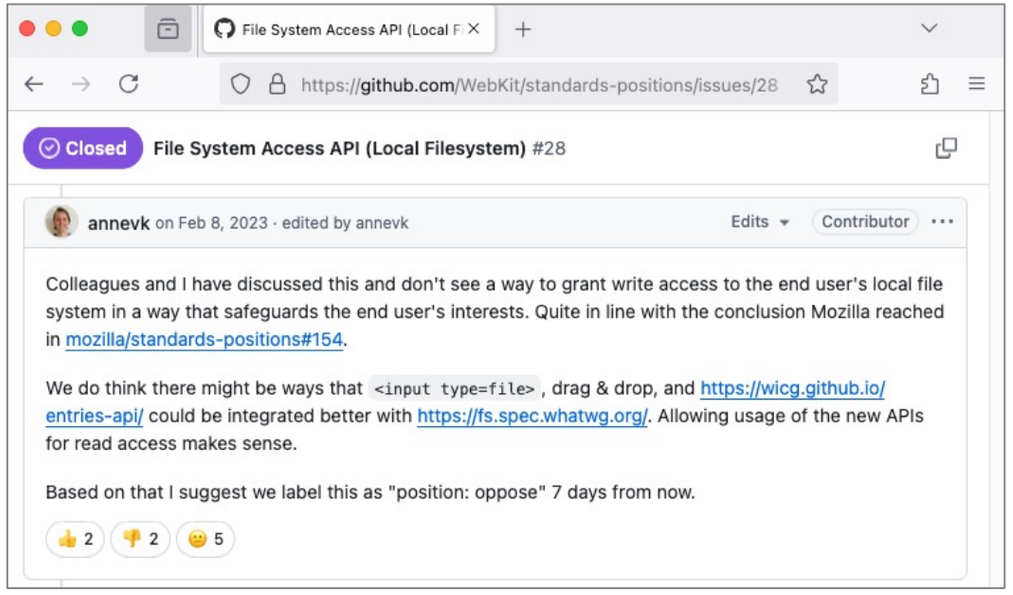
Overview of JSON-RPC Usage

File I/O within the Browser

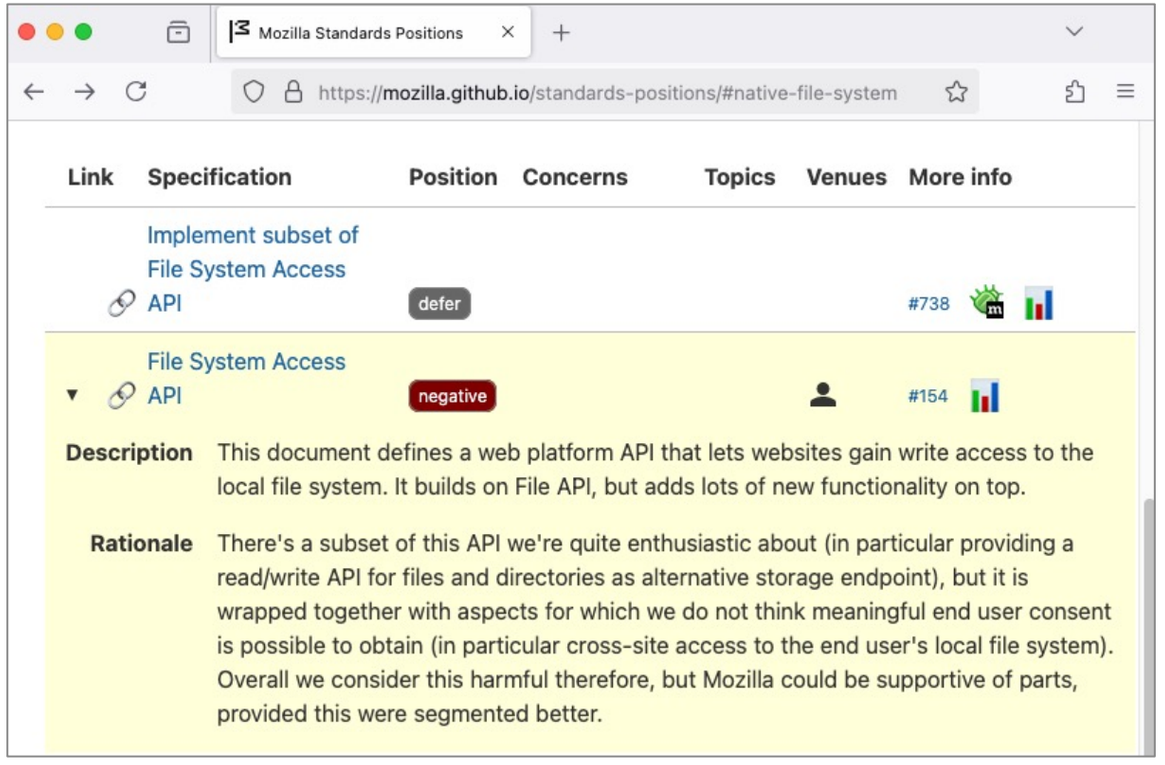


Overview of Browser File APIs

File System Access API – State in other Browsers



Position of WebKit on File System Access API[1]



Position of Firefox on File System Access API[2]

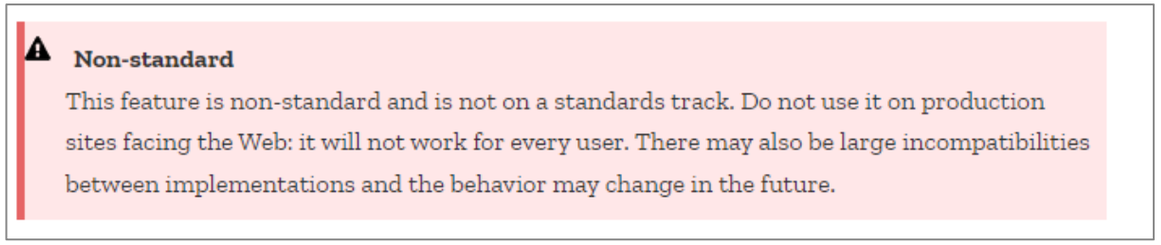
1. Github discussion about File System Access API in WebKit; https://github.com/WebKit/standards-positions/issues/28
2. Mozilla Standards Positions – File System Access API; https://mozilla.github.io/standards-positions/#native-file-system

File API / File Input Element

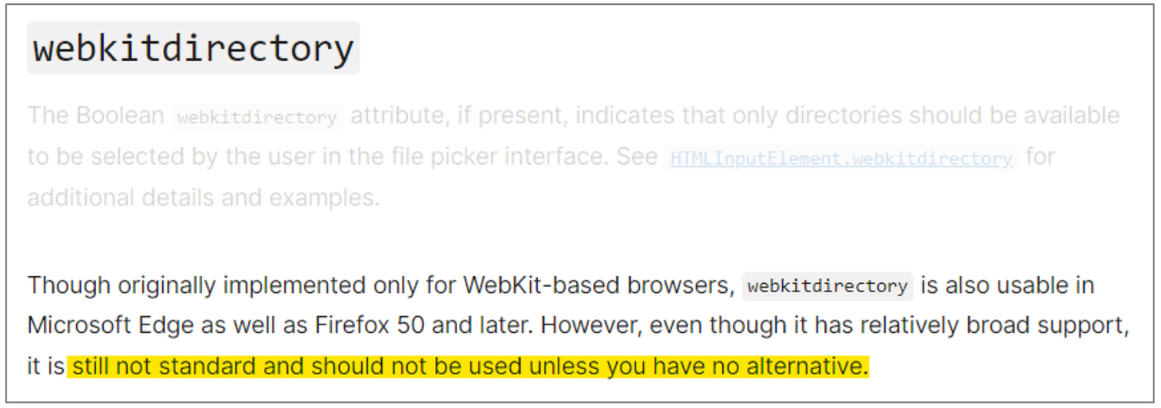
- Designed for single files.
- Lags if many (1000+) files are selected.

```
<input type="file" webkitdirectory />
<script>
  const picker = document.querySelector('input');
  picker.addEventListener('change', (e) => {
    for (let file of picker.files) {
      console.log(file.webkitRelativePath);
      // 'repo/README.md'
      // 'repo/.git/config'
      // 'repo/.git/HEAD'
      // 'repo/.git/objects/22/ee7b7ca19c25..'
      //  and many more ...
    }
  });
</script>
```

Simple Code Example of File Input Element



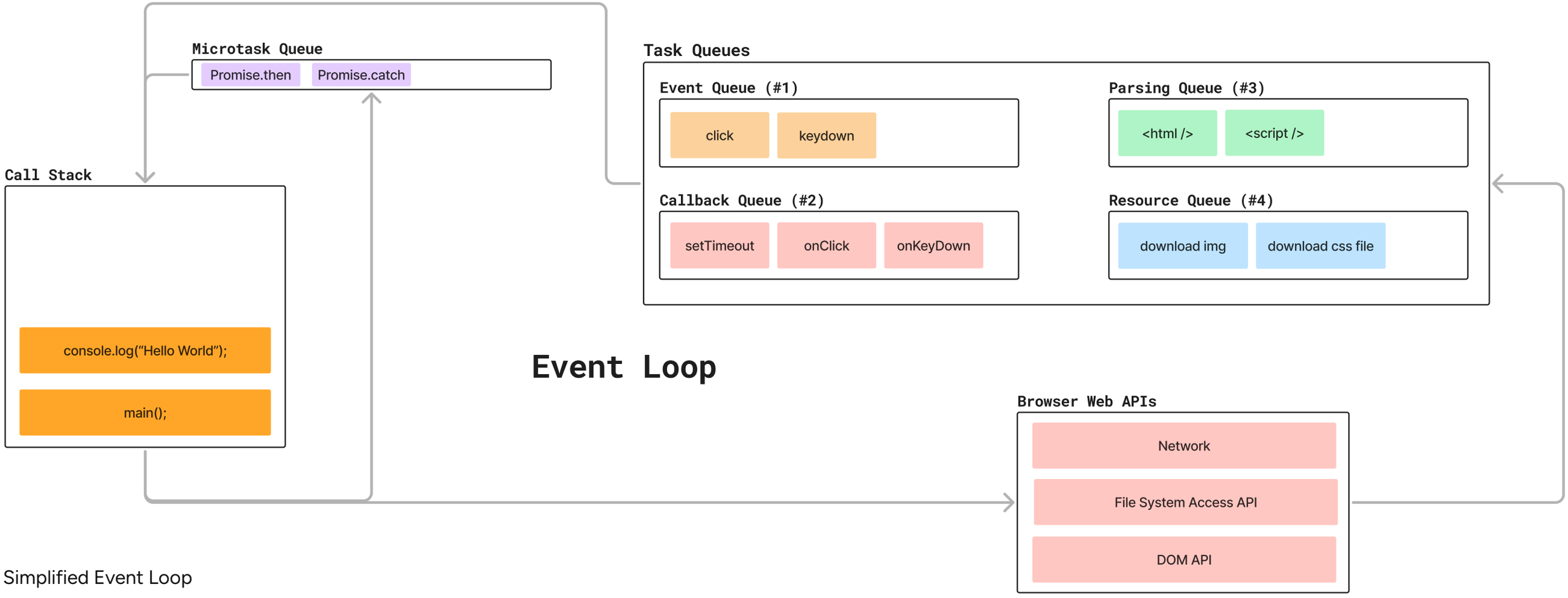
Warning about *webkitdirectory* attribute
[Screenshot taken from UDN web docs[1]]



Documentation of *webkitdirectory* attribute
[Screenshot taken from developer.mozilla.org[2]]

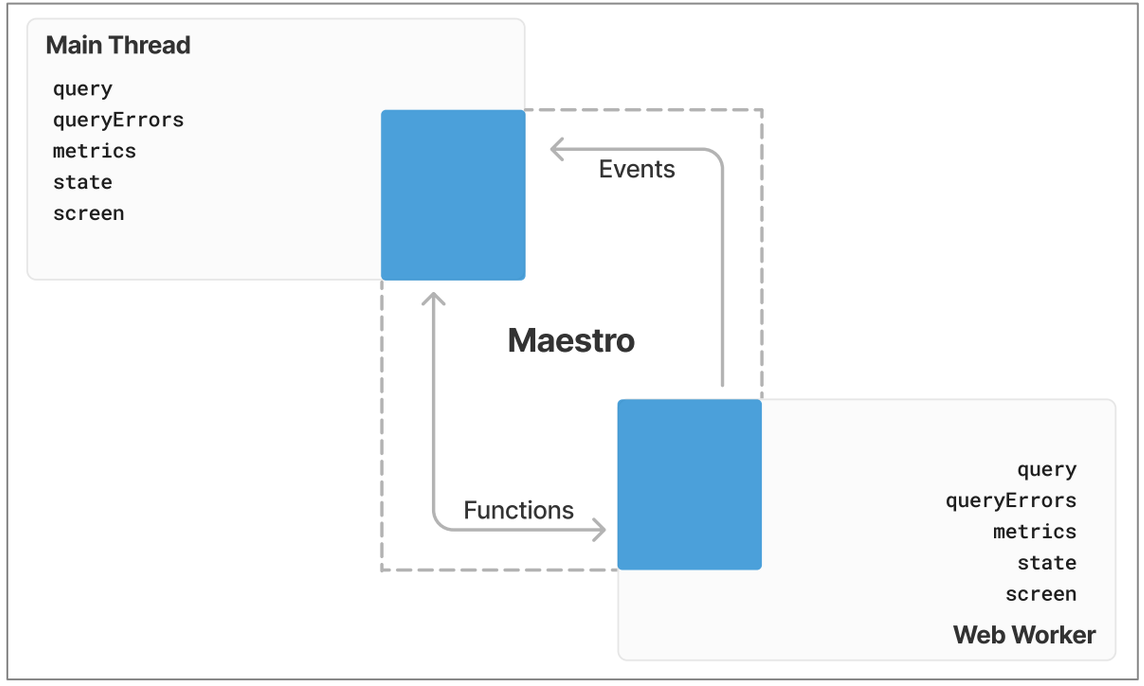
1. UDN web docs; HTMLInputElement.webkitdirectory; https://udn.realityripple.com/docs/Web/API/HTMLInputElement/webkitdirectory
2. Mozilla Documentation – File Input Element; https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file

Browser Event Loop

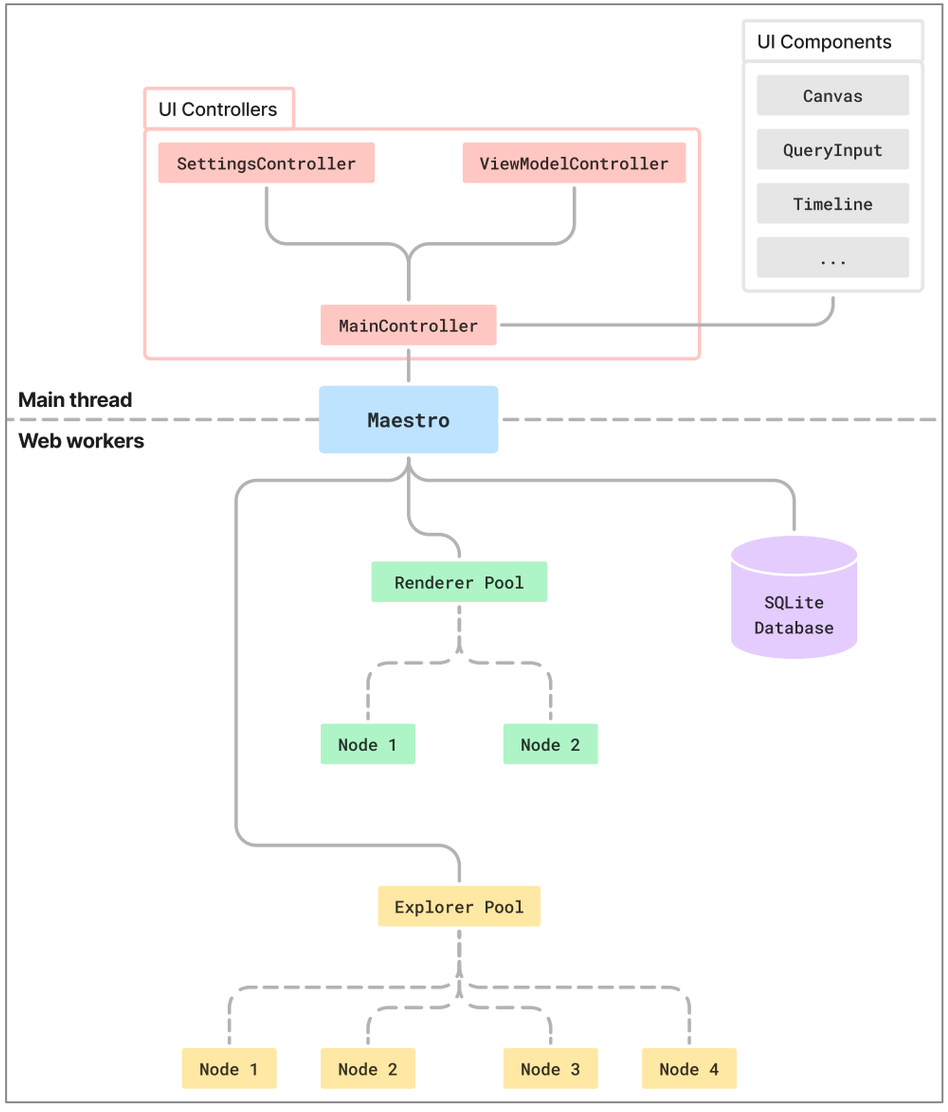


Simplified Event Loop

Architecture – Maestro



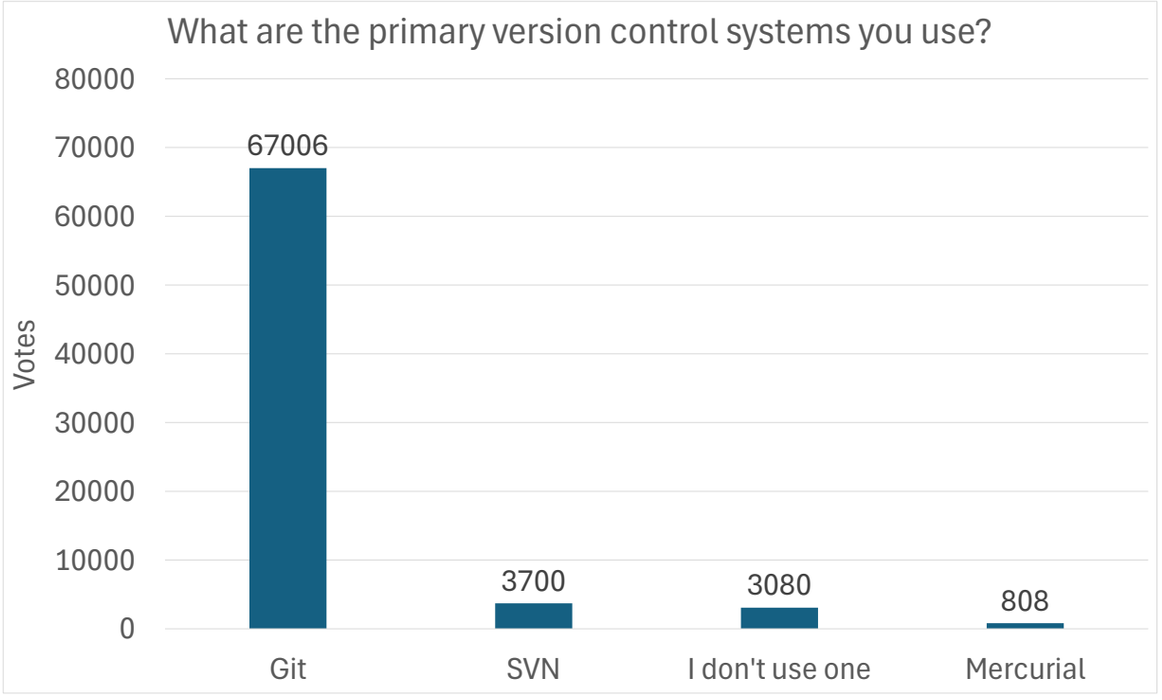
Maestro Controller



Gizual Architecture

Source-Code Version Control Systems

- Team collaboration.
- History of changes.
- Branching and merging for parallel development.
- Traceability of the author.



Survey results for the question "What are the primary version control systems you use?" from the Stack Overflow Developer Survey 2022. [based on Data of the Public 2022 Stack Overflow Developer Survey Results, licensed under the Open Database License (ODbL)][1]

1. Stack Overflow Developer Survey 2022
https://survey.stackoverflow.co/2022/