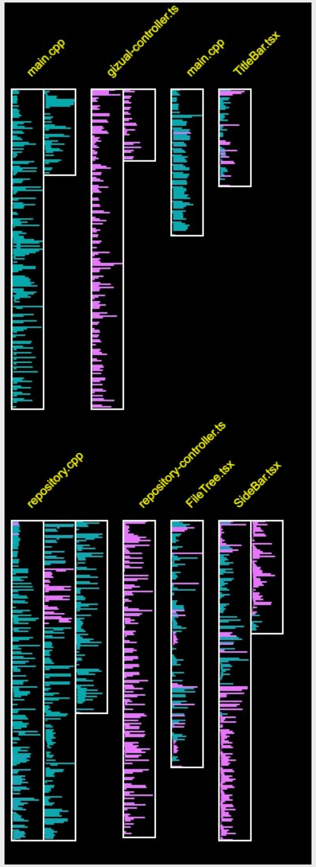
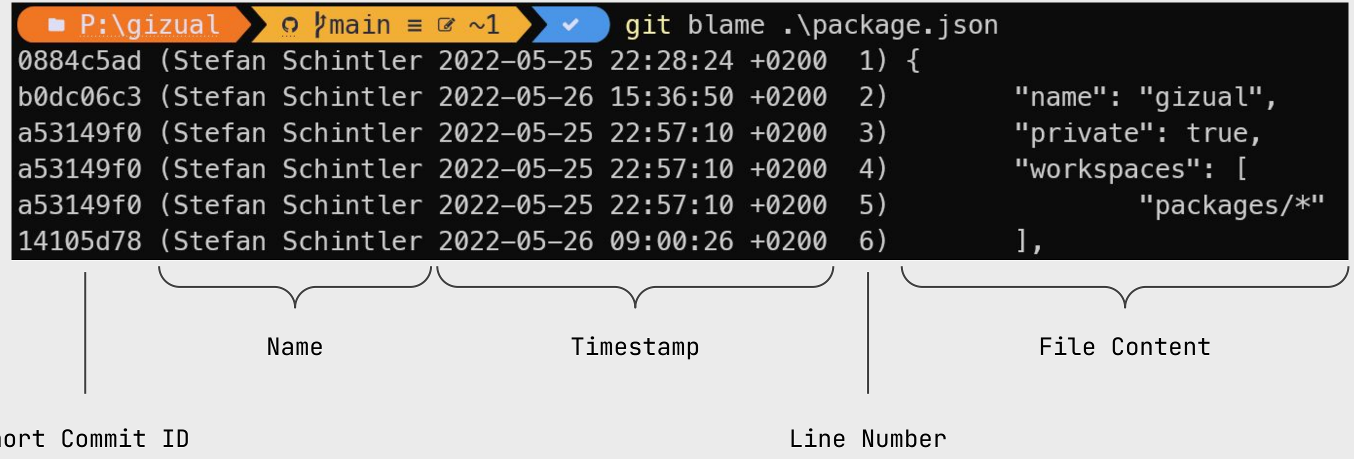
Gizual: Repository Visualisation for Git

```
// Project Presentation
const group1_members: [&str; 3] = [
    “Thomas Pinheiro de Souza”,
    “Stefan Schintler”,
    “Andreas Steinkellner”,
];
const presentation_date: &str = “2022-01-31”;
```



// Information Architecture and Web Usability, WS 2022

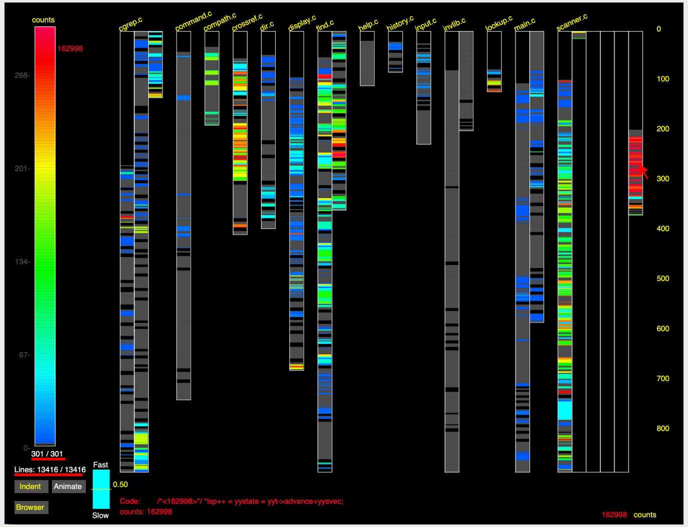
Git Blame



```
P:\gizual  main ≡ ~1  ✓  git blame .\package.json
0884c5ad (Stefan Schintler 2022-05-25 22:28:24 +0200  1) {
b0dc06c3 (Stefan Schintler 2022-05-26 15:36:50 +0200  2)         "name": "gizual",
a53149f0 (Stefan Schintler 2022-05-25 22:57:10 +0200  3)         "private": true,
a53149f0 (Stefan Schintler 2022-05-25 22:57:10 +0200  4)         "workspaces": [
a53149f0 (Stefan Schintler 2022-05-25 22:57:10 +0200  5)                 "packages/*"
14105d78 (Stefan Schintler 2022-05-26 09:00:26 +0200  6)         ],
```

Background

- Inspired by Seesoft[1]
- Visualise large repositories of code
- Easy visual comparison of:
  - File age
  - Line age
  - Author
  - File length
  - Change frequency
  - …



```
Code:    /*<162998>*/ *lsp++ = yystate = yyt->advance+yysvec;
counts: 162998
```

Image: Ball, Eick (1996): Software visualization in the Large.
“Program execution hotspots”

1: Eick, S. G., Steffen, J. L., & Sumner, E. E. (1992). Seesoft-a tool for visualizing line oriented software statistics

Previous Work (2022)

- Initial project work and proof of concept took place in the Information Visualisation course (SS22)[1].

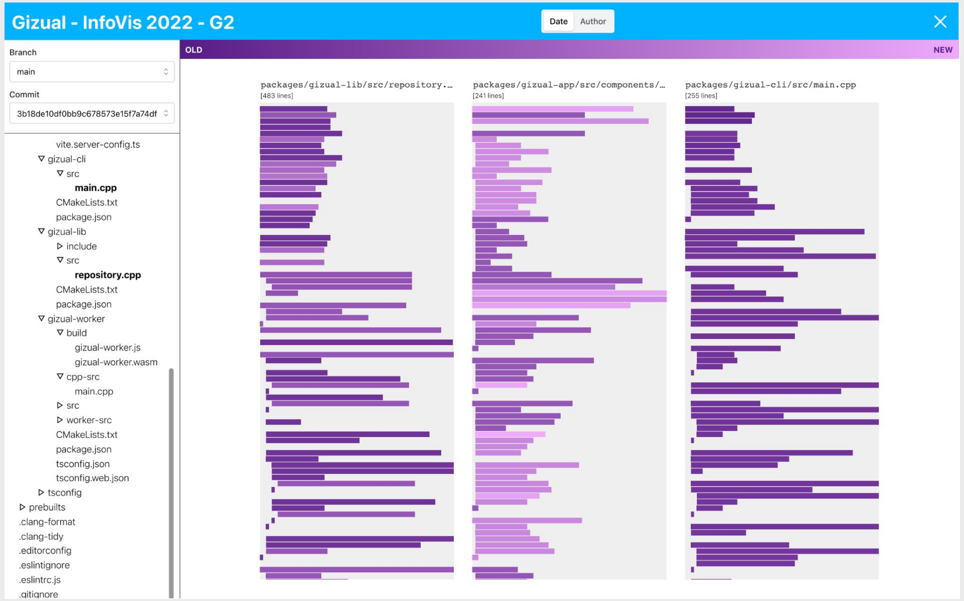


Image: Korduba, Schintler, Steinkellner (2022): Gizual: Repository Visualization for Git

1: Korduba, Schintler, Steinkellner (2022): Gizual: Repository Visualization for Git
https://courses.isds.tugraz.at/ivis/projects/ss2022/ivis-ss2022-g2-project-gizual-git-repo-vis.pdf

🔬

Toolchain

libgit2

https://libgit2.org/

- C implementation of the Git core methods.
- Minimal dependencies
- Cross-platform
- Permissive licensing (GPLv2)
- Used in production by many companies, including:
  - GitHub
  - GitLab

WebAssembly (WASM)

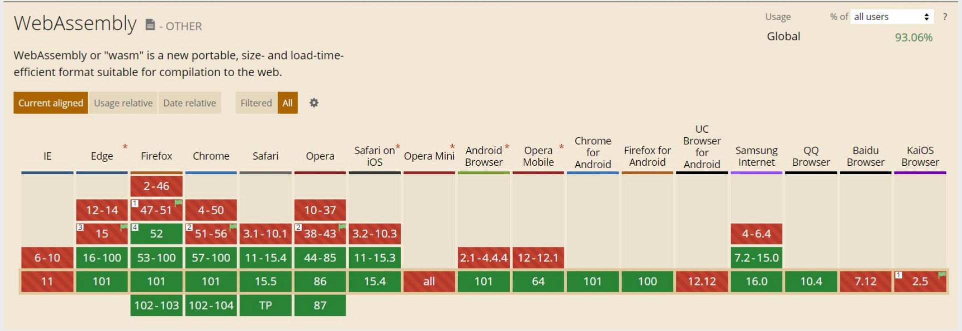
https://webassembly.org/

Facts:

- Modern binary instruction format
- Load-time-efficient virtual stack machine
- Sandboxed and memory-safe

Wasm allows to:

- Compile executables for the browser.
- Use existing native libraries within the browser.



WebAssembly - OTHER

Usage % of all users ?
Global 93.06%

WebAssembly or "wasm" is a new portable, size- and load-time-efficient format suitable for compilation to the web.

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
		🟥 2-46														
	🟥 12-14	🟥 [1] 47-51	🟥 4-50		🟥 10-37											
	🟥 [3] 15	🟩 [4] 52	🟥 [2] 51-56	🟥 3.1-10.1	🟥 [2] 38-43	🟥 3.2-10.3							🟥 4-6.4			
🟥 6-10	🟩 16-100	🟩 53-100	🟩 57-100	🟩 11-15.4	🟩 44-85	🟩 11-15.3		🟥 2.1-4.4.4	🟥 12-12.1				🟩 7.2-15.0			
🟥 11	🟩 101	🟩 101	🟩 101	🟩 15.5	🟩 86	🟩 15.4	🟥 all	🟩 101	🟩 64	🟩 101	🟩 100	🟥 12.12	🟩 16.0	🟩 10.4	🟥 7.12	🟥 [1] 2.5
		🟩 102-103	🟩 102-104	🟩 TP	🟩 87											

Screenshot taken from https://caniuse.com/wasm

Emscripten

https://github.com/emscripten-core/emscripten

- Compiler toolchain for WebAssembly
- Supports modern C and C++ (C++17 standard).
- Uses Clang[1] compiler and LLVM[2] toolchain.
- Ships with a pre-compiled standard library.
- Supports different forms of FileSystems (e.g. memoryFS).

1: Clang - https://clang.llvm.org/
2: LLVM - https://llvm.org/

Emscripten + libgit2 (1)

- No native ability to execute **http requests** from Wasm.
  + Fork libgit2 and apply patches[1].

- No native **filesystem** within Wasm.
  + Use Emscripten's memoryFS implementation.

- Libgit2's synchronous API **blocks** the **main thread** (unresponsive UI).
  + Run wasm module within a Web Worker[2].

1: Inspired by https://github.com/petersalomonsen/wasm-git
2: Web Worker - https://web.dev/workers-overview/

Emscripten + libgit2 (2)

- Unable to clone into the browser because of **CORS**[1].
  + Setup a proxy[2] on the server-side to relay requests.
- Cloning a large repository takes **too long.**
  + Use File System Access API as an alternative.
- Timestamps in int64 format, but Wasm is 32 bit.
  + Use of Emscripten's BigInt support for large numbers.

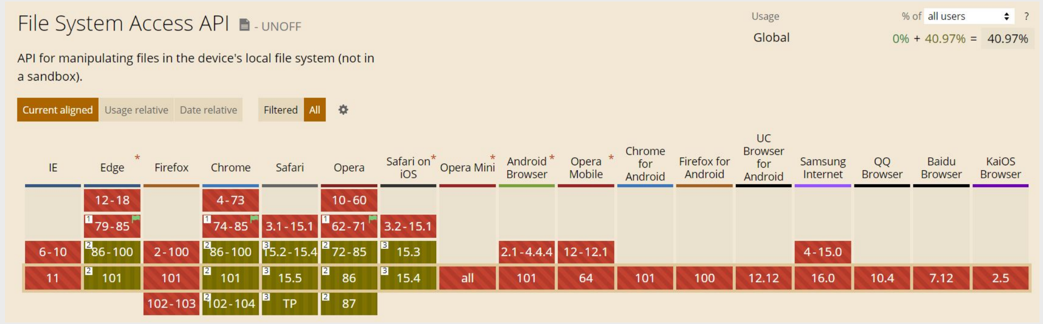
1: Cross-Origin Resource Sharing - https://web.dev/cross-origin-resource-sharing/
2: Inspired by https://github.com/isomorphic-git/cors-proxy

File System Access API

- Enables interaction with local files and folders.
- Users have to explicitly permit read and/or write access.

Allows us to:

- Copy local .git folder recursively into Emscripten's memoryFS.



File System Access API - UNOFF

API for manipulating files in the device's local file system (not in a sandbox).

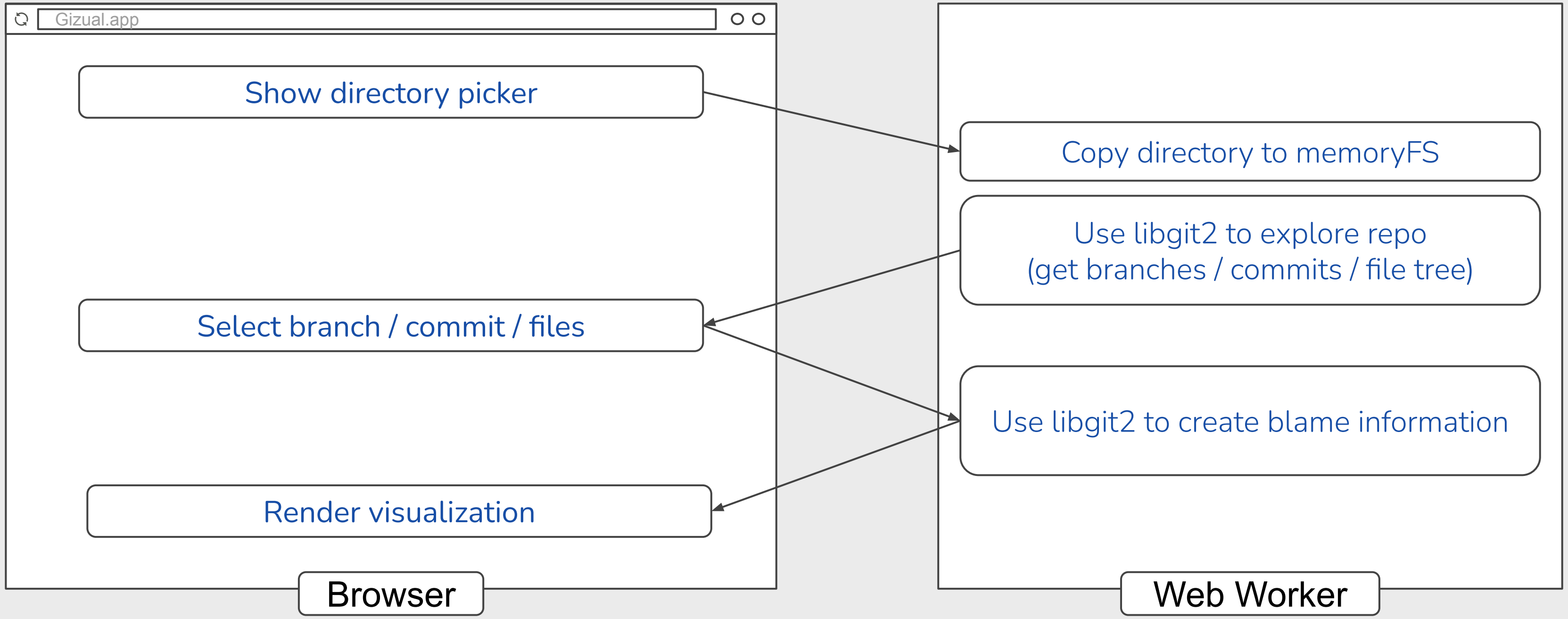
Usage: Global — % of all users ? — 0% + 40.97% = 40.97%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
	12-18		4-73		10-60											
	[1] 79-85		[1] 74-85	3.1-15.1	[1] 62-71	3.2-15.1										
6-10	[2] 86-100	2-100	[2] 86-100	[3] 15.2-15.4	[2] 72-85	[3] 15.3		2.1-4.4.4	12-12.1				4-15.0			
11	[2] 101	101	[2] 101	[3] 15.5	[2] 86	[3] 15.4	all	101	64	101	100	12.12	16.0	10.4	7.12	2.5
		102-103	[2] 102-104	[3] TP	[2] 87											

Screenshot taken from https://caniuse.com/native-filesystem-api

Current Approach


Gizual.app
Show directory picker
Select branch / commit / files
Render visualization
Browser
Copy directory to memoryFS
Use libgit2 to explore repo
(get branches / commits / file tree)
Use libgit2 to create blame information
Web Worker

Improved State Management

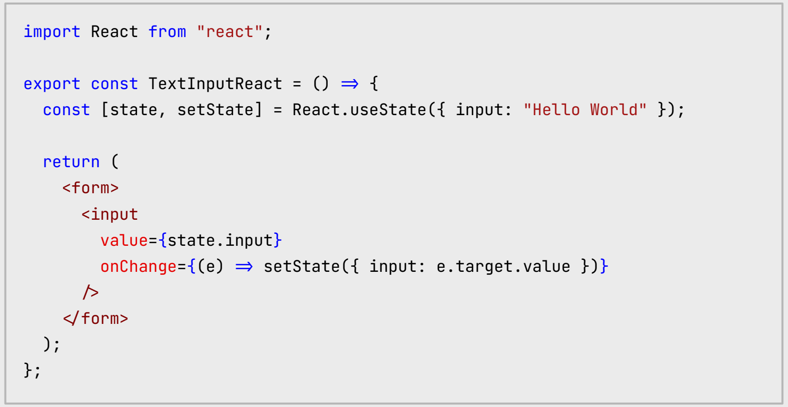
- Previous approach:
  - State management with nested React hooks.
  - Complicated to extend, tight coupling with view layer.
- New solution:
  - State management with MobX[1].
  - Controller instance manages application state.
  - State accessible from anywhere within the application.
  - Debuggable state through browser console.
  - File (blame result) caching.

1: MobX - https://mobx.js.org/

Native React State vs. MobX

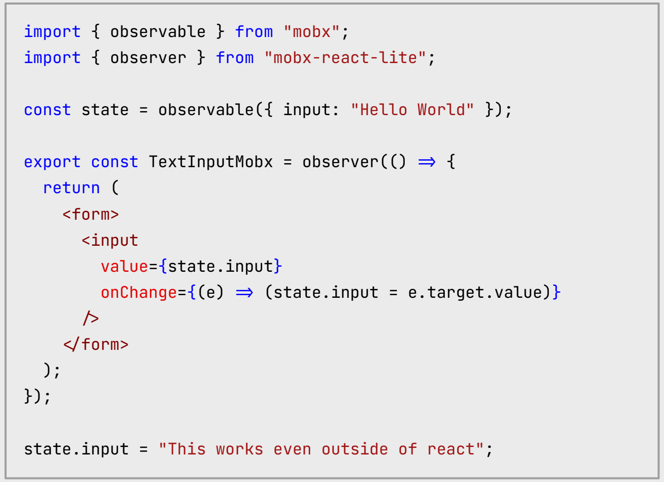
- Capture state between re-renders with useState.
- Core concept: Lifting state up (data flows down)

- Decoupled from the UI (view layer)
- Core concept: Observer automatically detects changes and triggers necessary rerenders.



```
import React from "react";

export const TextInputReact = () => {
  const [state, setState] = React.useState({ input: "Hello World" });

  return (
    <form>
      <input
        value={state.input}
        onChange={(e) => setState({ input: e.target.value })}
      />
    </form>
  );
};
```

Code example, simple text input with React state

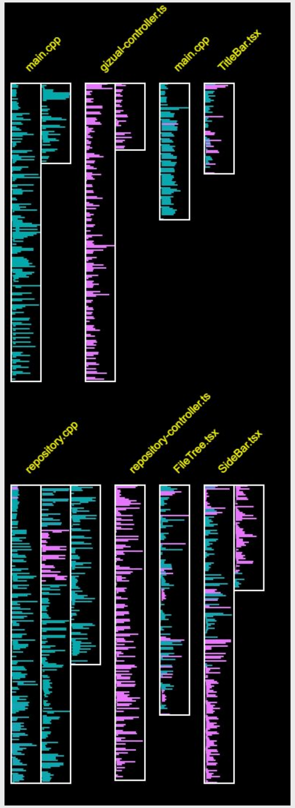


```
import { observable } from "mobx";
import { observer } from "mobx-react-lite";

const state = observable({ input: "Hello World" });

export const TextInputMobx = observer(() => {
  return (
    <form>
      <input
        value={state.input}
        onChange={(e) => (state.input = e.target.value)}
      />
    </form>
  );
});

state.input = "This works even outside of react";
```

Code example, simple text input with MobX state

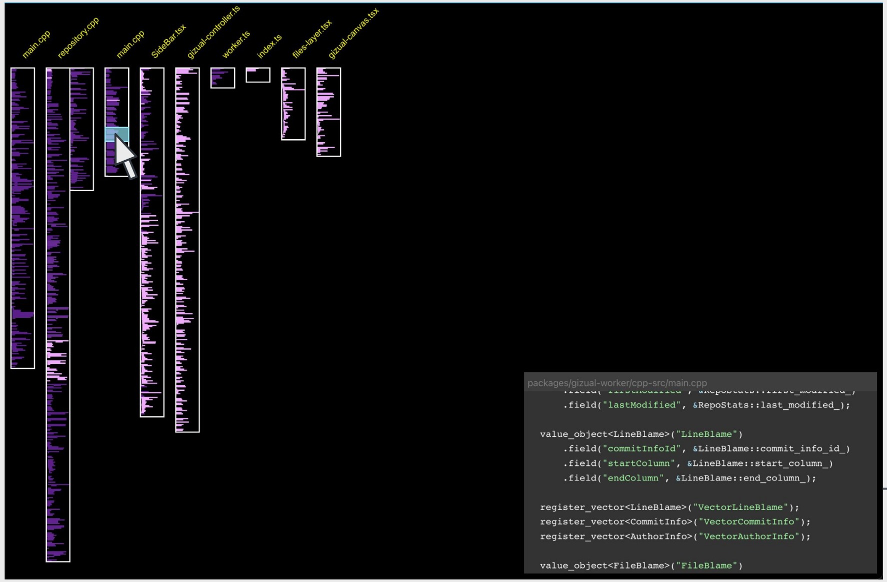
Visualization Canvas

- Rendered using a single 2D Canvas.
- Written in Konva.js[1].
- Design inspired by Seesoft.
- Automatic file wrapping based on available height.
- Automatic column width based on longest line of code.
- Horizontal scrolling



1: Konva.js - https://konvajs.org/

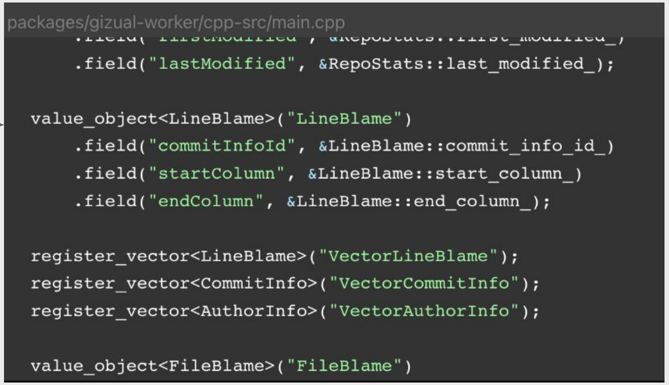
Code-Lens



packages/gizual-worker/cpp-src/main.cpp

```
        .field("lastModified", &RepoStats::last_modified_);

    value_object<LineBlame>("LineBlame")
        .field("commitInfoId", &LineBlame::commit_info_id_)
        .field("startColumn", &LineBlame::start_column_)
        .field("endColumn", &LineBlame::end_column_);

    register_vector<LineBlame>("VectorLineBlame");
    register_vector<CommitInfo>("VectorCommitInfo");
    register_vector<AuthorInfo>("VectorAuthorInfo");

    value_object<FileBlame>("FileBlame")
```

- Syntax highlighting with highlight.js[1].
- Floating window on top of canvas.
- Mouse indicator for displayed section.



packages/gizual-worker/cpp-src/main.cpp

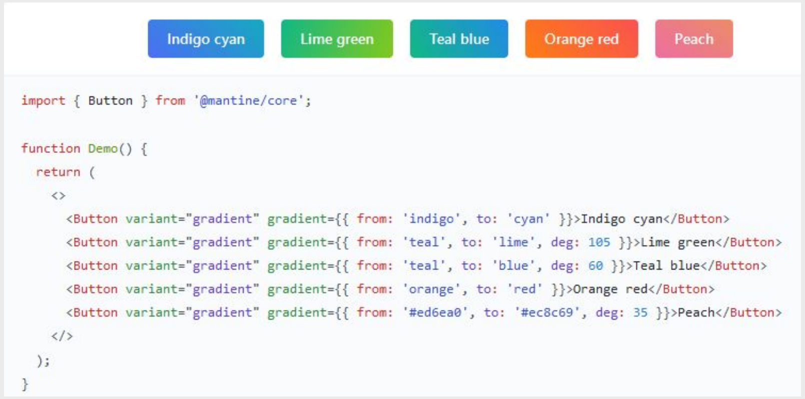
```
        .field("lastModified", &RepoStats::last_modified_);

    value_object<LineBlame>("LineBlame")
        .field("commitInfoId", &LineBlame::commit_info_id_)
        .field("startColumn", &LineBlame::start_column_)
        .field("endColumn", &LineBlame::end_column_);

    register_vector<LineBlame>("VectorLineBlame");
    register_vector<CommitInfo>("VectorCommitInfo");
    register_vector<AuthorInfo>("VectorAuthorInfo");

    value_object<FileBlame>("FileBlame")
```

1: highlight.js - https://highlightjs.org/

Mantine (Main UI)

https://mantine.dev/

- Open source components library.
- >100 customizable components & 40 hooks.
- Supports various different frameworks (Next.js, Remix, React, etc.).
- Based on Typescript.
- First stable release (1.0.0) in May 2021.



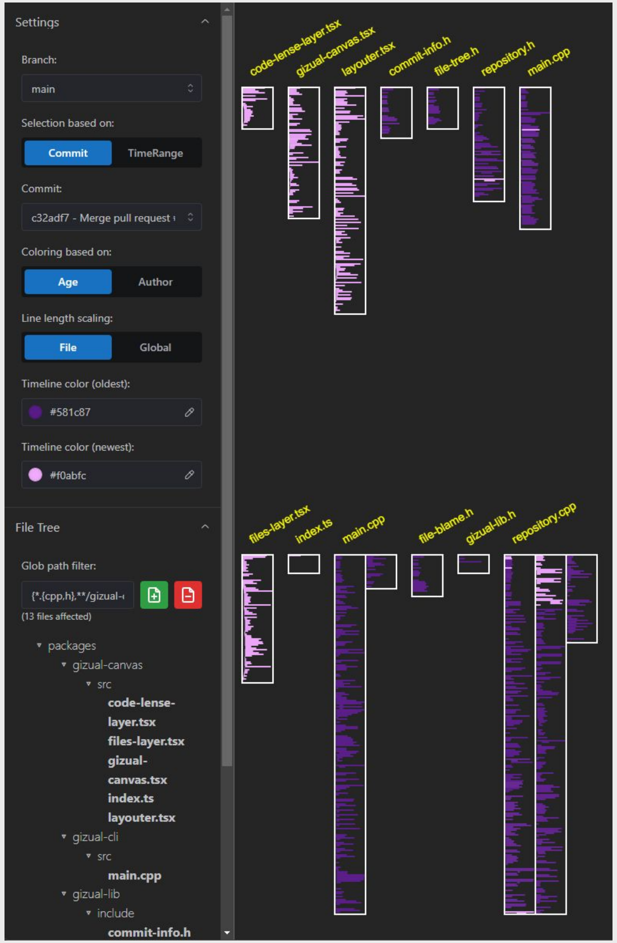
```
import { Button } from '@mantine/core';

function Demo() {
  return (
    <>
      <Button variant="gradient" gradient={{ from: 'indigo', to: 'cyan' }}>Indigo cyan</Button>
      <Button variant="gradient" gradient={{ from: 'teal', to: 'lime', deg: 105 }}>Lime green</Button>
      <Button variant="gradient" gradient={{ from: 'teal', to: 'blue', deg: 60 }}>Teal blue</Button>
      <Button variant="gradient" gradient={{ from: 'orange', to: 'red' }}>Orange red</Button>
      <Button variant="gradient" gradient={{ from: '#ed6ea0', to: '#ec8c69', deg: 35 }}>Peach</Button>
    </>
  );
}
```

Screenshot taken from https://mantine.dev/core/button/

💻 Features

Features

- Clone / use local repository
- Code-Lens
- Glob search
- User-definable options
  - Coloring
  - Time range selection
  - Line length scaling
- Horizontally infinite canvas



Settings

Branch: main

Selection based on: [✓ Commit] TimeRange

Commit: c32adf7 - Merge pull request

Coloring based on: [✓ Age] Author

Line length scaling: [✓ File] Global

Timeline color (oldest): #581c87

Timeline color (newest): #f0abfc

File Tree

Glob path filter: {*.{cpp,h},**/gizual-

(13 files affected)

- packages
  - gizual-canvas
    - src
      - code-lense-layer.tsx
      - files-layer.tsx
      - gizual-canvas.tsx
      - index.ts
      - layouter.tsx
  - gizual-cli
    - src
      - main.cpp
  - gizual-lib
    - include
      - commit-info.h

Supported Modes

Selection Mode / Coloring	**Specific Commit** (User selects a commit)	**Time Range** (User selects a time range, last commit within time range is pre-selected)
Author (per line of code)	Canvas colored based on author	Canvas colored based on author, Lines before specified range uncolored
Age (per line of code)	Canvas colored based on age	Canvas colored based on age, Lines before specified range uncolored

Live Demo

(or try it yourself at https://gizual.xyz)

🐛

Limitations

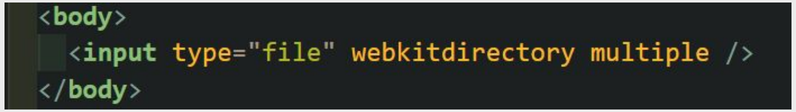
Static Deployment vs. Docker

	Static Deployment	Docker
Deployment Complexity	😁👌 Simple webspace	🙇😅 Linux server with SSH, Docker and root access
Feature: git pull	Impossible (missing git protocol proxy server), blocked by CORS[1]	Possible for repositories up to ~20MB

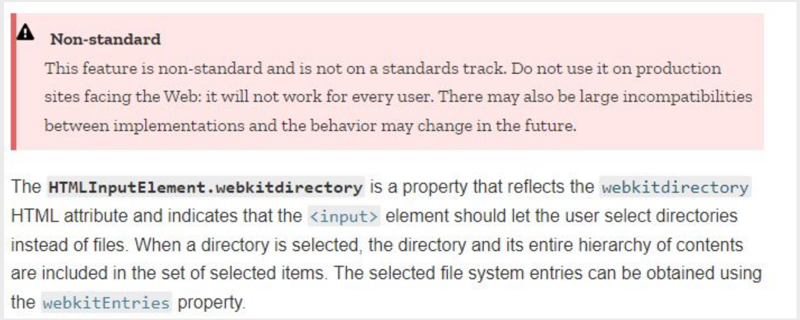
1: CORS - Cross-Origin Resource Sharing

Firefox Compatibility

- No File System Access API, but …
  - Non-standard legacy option: `webkitdirectory`
  - Works, but may be slow and laggy on large repos.
  - Returns FileHandles for each file within the folder.
  - Has some new performance limitations.
- Requires use of an older Web Worker Format called “iife” instead of the modern “es” format.



```
<body>
  <input type="file" webkitdirectory multiple />
</body>
```

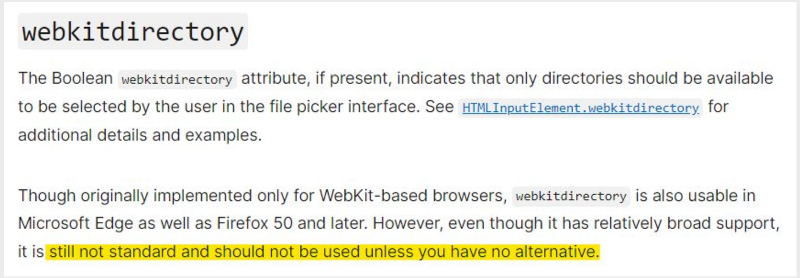


Non-standard

This feature is non-standard and is not on a standards track. Do not use it on production sites facing the Web: it will not work for every user. There may also be large incompatibilities between implementations and the behavior may change in the future.

The **`HTMLInputElement.webkitdirectory`** is a property that reflects the `webkitdirectory` HTML attribute and indicates that the `<input>` element should let the user select directories instead of files. When a directory is selected, the directory and its entire hierarchy of contents are included in the set of selected items. The selected file system entries can be obtained using the `webkitEntries` property.

Screenshot taken from:
https://udn.realityripple.com/docs/Web/API/HTMLInputElement/webkitdirectory



`webkitdirectory`

The Boolean `webkitdirectory` attribute, if present, indicates that only directories should be available to be selected by the user in the file picker interface. See `HTMLInputElement.webkitdirectory` for additional details and examples.

Though originally implemented only for WebKit-based browsers, `webkitdirectory` is also usable in Microsoft Edge as well as Firefox 50 and later. However, even though it has relatively broad support, it is still not standard and should not be used unless you have no alternative.

Screenshot taken from:
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file

Issue: Blame Performance

- Libgit2 file blame is slow for some files - open issue on: https://github.com/libgit2/libgit2/issues/3027
- Core git solved blame performance problems a while back, but due to lack of funding of a single contributor these changes are deliberately not permitted to be ported to libgit2.
- Quote from the git contributors message board:

> “So the price tag for letting the finished git-blame work (I've found a few more optimizations making it more worthwhile) get relicensed under the libgit2 licensing scheme would be in the order of €10000. It would take a rather good salaried programmer to reproduce what I'm doing right now for the same price tag, and since my work will be available in Git proper under the GPLv2 before anybody has to make any decision, there is no uncertainty about exactly what people will be getting.”

Andreas Ericsson <ae@op5.se>, 2014 - git@vger.kernel.org list mirror, accessible at: https://public-inbox.org/git/87vbwwxfol.fsf@fencepost.gnu.org/

Performance

- Performance varies greatly between browsers, release versions, repositories and files.
- Example (git blame performance on the same file):

```
○  Firefox  107: 581.669ms
○  Firefox  108:   4.726ms
○  Chrome   108:  16.380ms
○  Chrome   109:  16.392ms
```

Future Work

- Performance optimisations
- Different visualisation options
- Mergeable authors
- Smart selection of files
- Custom timeline for range selection
- Custom canvas settings (file width, maximum length, …)
- Infinite zoom into files
- Scaleable / Moveable Code-Lens

👋🎤

Thanks for your attention!