fathyb / carbonyl Public

Chromium running inside your terminal

BSD-3-Clause license

14.2k stars 279 forks Branches Tags Activity

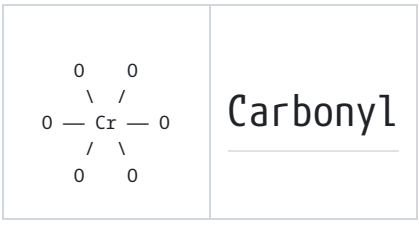
Star Notifications

Code Issues 65 Pull requests 3 Discussions Projects Security Insights

main 1 Branch 2 Tags Go to file Go to file Code

Stephen-Seo last year

.cargo	feat: linux support	last year
.github	chore: add funding.yml	last year
.refloat	fix(ci): package proper library in binari···	last year
chromium	feat(browser): introduce quadrant renderi···	last year
scripts	feat(browser): introduce quadrant renderi···	last year
src	fix(renderer): fix idling CPU usage (#126)	last year
.dockerignore	feat: better true color detection	last year
.gitignore	chore(release): 0.0.2	last year
.gitmodules	docs(readme): fix build instructions (#15)	last year
Cargo.lock	chore(release): 0.0.3	last year
Cargo.toml	chore(release): 0.0.3	last year
Dockerfile	feat(browser): introduce quadrant renderi···	last year
build.rs	fix(build): link to Chromium sysroot libs···	last year
changelog.md	chore(release): 0.0.3	last year
cliff.toml	docs(changelog): fix commit_preprocessors···	last year
license.md	docs: add license	last year
package.json	chore(release): 0.0.3	last year
readme.md	docs(readme): update download links	last year



Carbonyl is a Chromium based browser built to run in a terminal. Read the blog post.

It supports pretty much all Web APIs including WebGL, WebGPU, audio and video playback, animations, etc..

It's snappy, starts in less than a second, runs at 60 FPS, and idles at 0% CPU usage. It does not require a window server (i.e. works in a safe-mode console), and even runs through SSH.

Carbonyl originally started as html2svg and is now the runtime behind it.

Usage

> Carbonyl on Linux without Docker requires the same dependencies as Chromium.

Docker

```
$ docker run --rm -ti fathyb/carbonyl https://youtube.com
```

npm

```
$ npm install --global carbonyl
$ carbonyl https://github.com
```

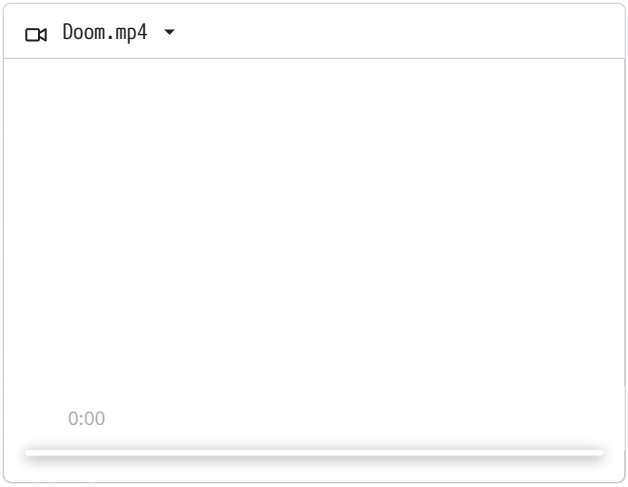
Binaries

- macOS amd64
- macOS arm64
- Linux amd64
- Linux arm64

Demo

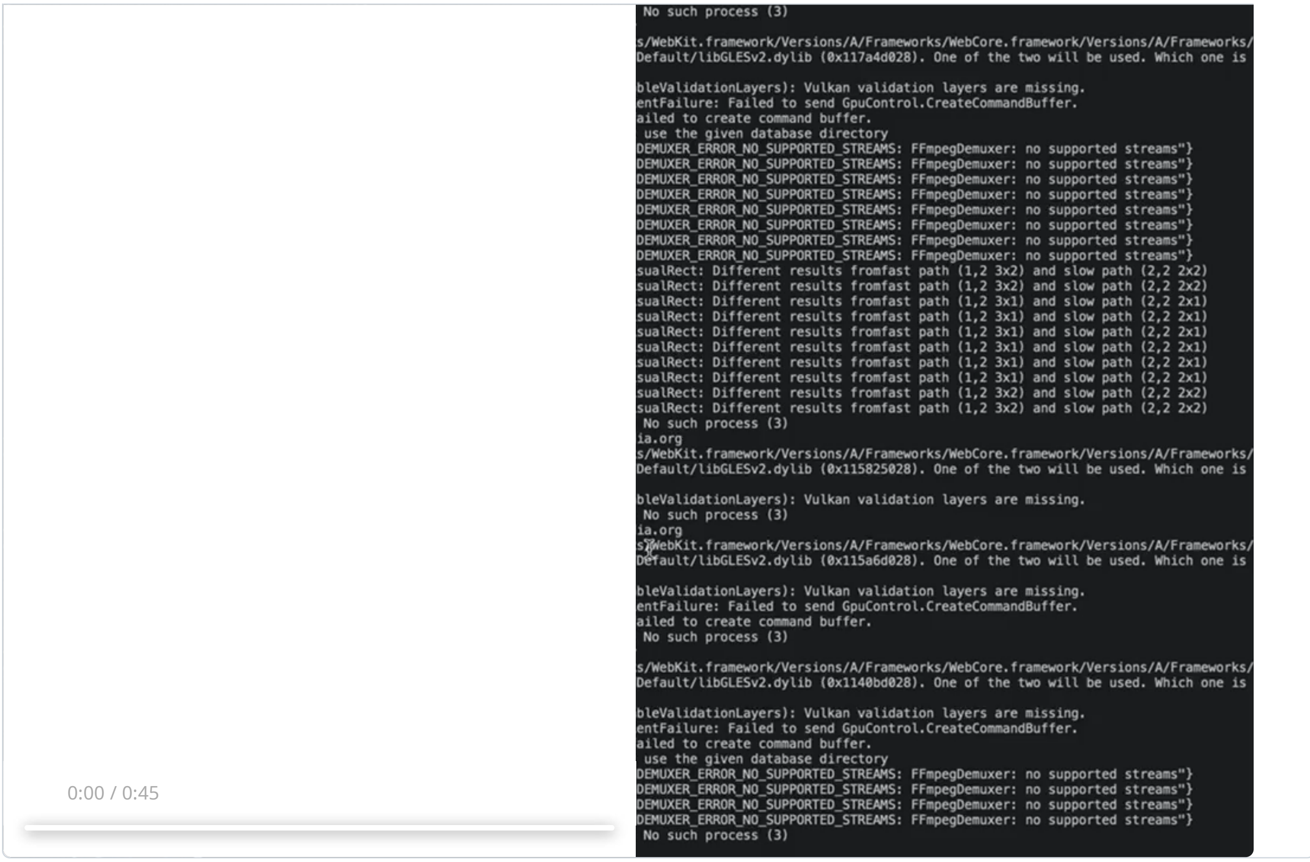
Wikipedia.mp4

Doom.mp4



YouTube.mp4

README BSD-3-Clause license



Known issues

- Fullscreen mode not supported yet

Comparisons

Lynx

Lynx is the original terminal web browser, and the oldest one still maintained.

Pros

- When it understands a page, Lynx has the best layout, fully optimized for the terminal

Cons

> Some might sound like pluses, but Browsh and Carbonyl let you disable most of those if you'd like

- Does not support a lot of modern web standards
- Cannot run JavaScript/WebAssembly
- Cannot view or play media (audio, video, DOOM)

Browsh

Browsh is the original "normal browser into a terminal" project. It starts Firefox in headless mode and connects to it through an automation protocol.

Pro

- It's easier to update the underlying browser: just update Firefox
- This makes development easier: just install Firefox and compile the Go code in a few seconds
- As of today, Browsh supports extensions while Carbonyl doesn't, although it's on our roadmap

Cons

- It runs slower and requires more resources than Carbonyl. 50x more CPU power is needed for the same content in average, that's because Carbonyl does not downscale or copy the window framebuffer, it natively renders to the terminal resolution.
- It uses custom stylesheets to fix the layout, which is less reliable than Carbonyl's changes to its HTML engine (Blink).

Operating System Support

As far as tested, the operating systems under are supported:

- Linux (Debian, Ubuntu and Arch tested)
- MacOS
- Windows 11 and WSL

Contributing

Carbonyl is split in two parts: the "core" which is built into a shared library (`libcarbonyl`), and the "runtime" which dynamically loads the core (`carbonyl` executable).

The core is written in Rust and takes a few seconds to build from scratch. The runtime is a modified version of the Chromium headless shell and takes more than an hour to build from scratch.

If you're just making changes to the Rust code, build `libcarbonyl` and replace it in a release version of Carbonyl.

Core

```
$ cargo build
```

Runtime

Few notes:

- Building the runtime is almost the same as building Chromium with extra steps to patch and bundle the Rust library. Scripts in the `scripts/` directory are simple wrappers around `gn` , `ninja` , etc..
- Building Chromium for arm64 on Linux requires an amd64 processor
- Carbonyl is only tested on Linux and macOS, other platforms likely require code changes to Chromium
- Chromium is huge and takes a long time to build, making your computer mostly unresponsive. An 8-core CPU such as an M1 Max or an i9 9900k with 10 Gbps fiber takes around ~1 hour to fetch and build. It requires around 100 GB of disk space.

Fetch

> Fetch Chromium's code.

```
$ ./scripts/gclient.sh sync
```

Apply patches

> Any changes made to Chromium will be reverted, make sure to save any changes you made.

```
$ ./scripts/patches.sh apply
```

Configure

```
$ ./scripts/gn.sh args out/Default
```

> `Default` is the target name, you can use multiple ones and pick any name you'd like, i.e.:
>
> ```
> $ ./scripts/gn.sh args out/release
> $ ./scripts/gn.sh args out/debug
> # or if you'd like to build a multi-platform image
> $ ./scripts/gn.sh args out/arm64
> $ ./scripts/gn.sh args out/amd64
> ```

When prompted, enter the following arguments:

```
import("//carbonyl/src/browser/args.gn")

# uncomment this to build for arm64
# target_cpu = "arm64"
```

```
# comment this to disable ccache
cc_wrapper = "env CCACHE_SLOPPINESS=time_macros ccache"

# comment this for a debug build
is_debug = false
symbol_level = 0
is_official_build = true
```

Build binaries

```
$ ./scripts/build.sh Default
```

This should produce the following outputs:

- out/Default/headless_shell : browser binary
- out/Default/icudtl.dat
- out/Default/libEGL.so
- out/Default/libGLESv2.so
- out/Default/v8_context_snapshot.bin

Build Docker image

```
# Build arm64 Docker image using binaries from the Default target
$ ./scripts/docker-build.sh Default arm64
# Build amd64 Docker image using binaries from the Default target
$ ./scripts/docker-build.sh Default amd64
```

Run

```
$ ./scripts/run.sh Default https://wikipedia.org
```

Releases 2

0.0.3 Latest
on Feb 18, 2023

+ 1 release

Sponsor this project

Contributors 11

Languages

Rust 68.7% C++ 10.7% JavaScript 9.8% Shell 8.2% C 1.3% Dockerfile 1.1% Python 0.2%