



What's new with the Firefox Profiler

Nazim Can Altinova

<https://share.firefox.dev/profiler-whatsnew-fosdem-2023>



Table of contents

- Intro
 - What is a profiler?
 - What is Firefox Profiler?
- Importers from other profilers
- Other tools that use the Firefox Profiler
- New features & UI improvements
 - Power profiling
 - Source code view & inline call stacks
 - And many more!

profiler.firefox.com

github.com/firefox-devtools/profiler/

What is a profiler?

**Helps developers to analyze
performance issues.**



It gives insight and clues.

This is detective work.



What is the Firefox Profiler

Statistical profiler with additional data

**Main data sources:
Sampling + Markers**



Introduction to Firefox Profiler talk:

By Julien Wajsberg
JavaScript devroom
Sunday, 16:30 - 16:55



**Importers from other
profilers**

Currently supported profile formats

1. Chrome Performance Tab – Trace Event Format
2. Linux perf script
3. ART Trace
4. Valgrind DHAT




**All you need to do is drag and
drop the file into
profiler.firefox.com**



Firefox Profiler — Web app for Firefox performance analysis

Capture a performance profile. Analyze it. Share it. Make the web faster.



Documentation

To start profiling, click on the profiling button, or use the keyboard shortcuts. The icon is blue when a profile is recording. Hit **Capture** to load the data into profiler.firefox.com.

Ctrl + Shift + 1 Stop and start profiling

Ctrl + Shift + 2 Capture and load profile

You can also profile Firefox for Android. For more information, please consult this documentation: [Profiling Firefox for Android directly on device.](#)

Load existing profiles

You can **drag and drop** a profile file here to load it, or:

[Load a profile from file](#) [Load a profile from a URL](#)

Your recent uploaded recordings

1:11 AM **Profile #68rf9e** (1.0s)
🔄 Firefox 111 🍏 macOS 13.1.0

[Legal](#) [Privacy](#) [Cookies](#) [English \(US\)](#)



ETW importer

By Jeff Muizelaar

<https://github.com/jrmuizel/etw-profiler/tree/main/etw-gecko>



Other tools that use the Firefox Profiler

Java JFR Profiler

By Johannes Bechberger (SAP)

The screenshot shows the IntelliJ IDEA IDE with the Java JFR Profiler plugin active. The interface is divided into several panels:

- Project View:** Shows the project structure with files like `flight.json`, `flight.tar.gz`, and `profile.jfr`.
- Call Tree:** Displays a hierarchical view of the call stack. The root node is `com.github.partitioned.test.Main`. The tree shows a recursive call to `Main.fib(int): int`. The table below summarizes the call tree data:

Total (samples)	Self	Children	Method
100%	19	—	<code>Main.main(String[])</code>
53%	10	3	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
37%	7	—	<code>Main.fib(int): int</code>
26%	5	—	<code>Main\$Bla.fib(int): int</code>
21%	4	—	<code>Main\$1.fib(int): int</code>

- Flame Graph:** A horizontal bar chart showing the execution time of the function. The x-axis represents time in seconds, ranging from 0.00s to 0.60s.
- Call Node Details:** Shows the details for the selected node `Main.fib(int): int`. It includes the following information:
 - Traced running time: 292ms
 - Traced self time: —
 - Running samples: 37%
 - Self samples: —
 - Categories: Running sample count
 - Java: 100%
- Code Editor:** Displays the source code of the `Main.fib(int): int` method, showing a recursive Fibonacci function.

<https://plugins.jetbrains.com/plugin/20937-java-jfr-profiler>



Firefox Profiler beyond the web

By Johannes Bechberger (SAP)

Mozilla devroom

Saturday, 17:30 - 18:00



**New features
&
UI improvements**

Power profiling



Profiler Settings



Recordings launch profiler.firefox.com in a new tab. All data is stored local you can choose to upload it for sharing.

- Web Developer**
Recommended preset for most web app debugging, with low overhead.
- Nightly**
Recommended preset for profiling Nightly.
- Graphics**
Preset for investigating graphics bugs in Nightly.
- Media**
Preset for investigating audio and video bugs in Nightly.
- Networking**
Preset for investigating networking bugs in Nightly.
- Power**
Preset for investigating power use bugs in Nightly, with low overhead.
- Custom**

Full Settings

Firefox Profiler

Settings

Power

Preset for investigating power use bugs in Nightly, with low overhead.

[Edit Settings...](#)

Start Recording

^⬆️1



5 / 380 tracks

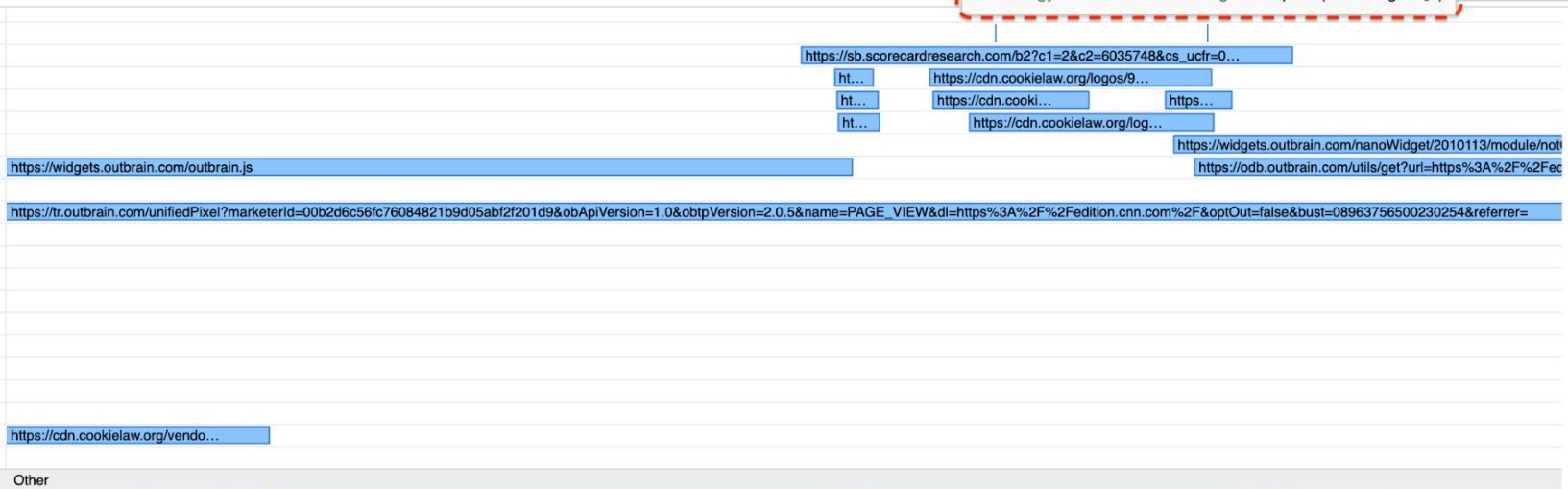
3.95s 4.00s 4.05s 4.10s 4.15s 4.20s 4.25s 4.30s 4.35s 4.40s 4.45s 4.50s 4.55s 4.60s 4.65s 4.70s 4.75s 4.80s 4.85s



Marker Chart Marker Table Network

Power: 2.68 W
 Energy used in the current selection: 63 μWh (0.028 mg CO₂e)
 Energy used in the visible range: 168 μWh (0.074 mg CO₂e)

Load
Unload
Network Requests



With CO₂ emission information

Power: 2.68 W

Energy used in the current selection: 63 μ Wh (0.028 mg CO₂e)

Energy used in the visible range: 168 μ Wh (0.074 mg CO₂e)

Thanks to **Chris Adams** and **Fershad** from **The Green Web Foundation**.



Power profiling with the Firefox Profiler talk:

By Florian Quèze

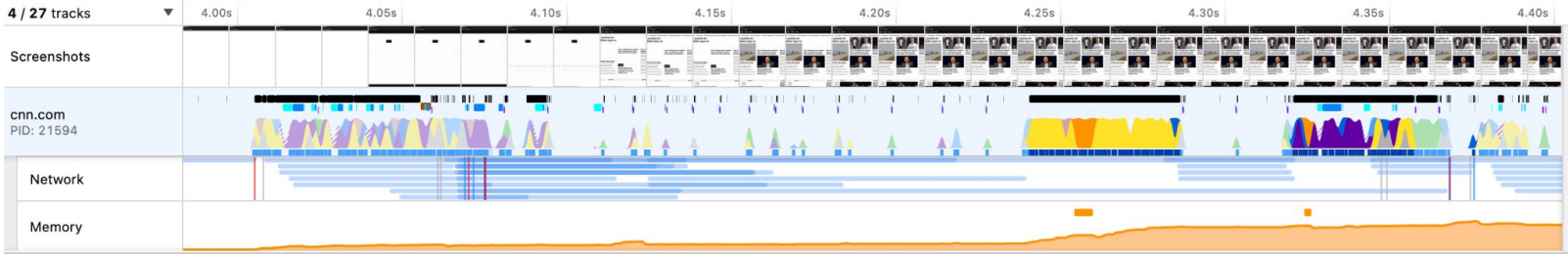
Energy devroom

Saturday, 17:30 - 18:00



Source code view & inline call stacks





Call Tree Flame Graph Stack Chart Marker Chart Marker Table Network

All stacks JavaScript Native Invert call stack Filter stacks: Enter filter terms

Complete "https://cnn.com" > Drop: __synch_cwait

Total (samples)	Self	
38%	82	in mozilla::dom::PromiseJobCallback::Call(mozilla::ErrorResult&, char const*, mozilla::dom::Callbac
38%	82	in mozilla::dom::PromiseJobCallback::Call(mozilla::dom::BindingCallContext&, JS::Handle<JS::Va
38%	82	JS::Call(JSContext*, JS::Handle<JS::Value>, JS::Handle<JS::Value>, JS::Handle<JS::Value>, JS::Handle<JS::Va
38%	82	JS::Call(JSContext*, JS::Handle<JS::Value>, JS::Handle<JS::Value>, js::AnyInvokeArgs const&,
38%	82	in InternalCall(JSContext*, js::AnyInvokeArgs const&, js::CallReason) js/src/vm/Interpreter.
38%	82	in js::InternalCallOrConstruct(JSContext*, JS::CallArgs const&, js::MaybeConstruct, js::Ca
38%	82	in CallJSNative(JSContext*, bool (*)(JSContext*, unsigned int, JS::Value*), js::CallReaso
38%	82	in PromiseReactionJob(JSContext*, unsigned int, JS::Value*) js/src/builtin/Promise.cpp
38%	82	in js::Call(JSContext*, JS::Handle<JS::Value>, JS::Handle<JS::Value>, JS::Handle<JS::Va
38%	82	js::RunScript
21%	46	https://cdn.cookieclaw.org/scripttemplates/202211.2.0/otBannerSdk.js:7:493
16%	35	in js::Call(JSContext*, JS::Handle<JS::Value>, JS::Handle<JS::Value>, js::AnyInvc

js::Call
js/src/vm/Interpreter.h

Call node details

Traced running time 81.9ms
Traced self time —
Running samples 38% 82
Self samples —

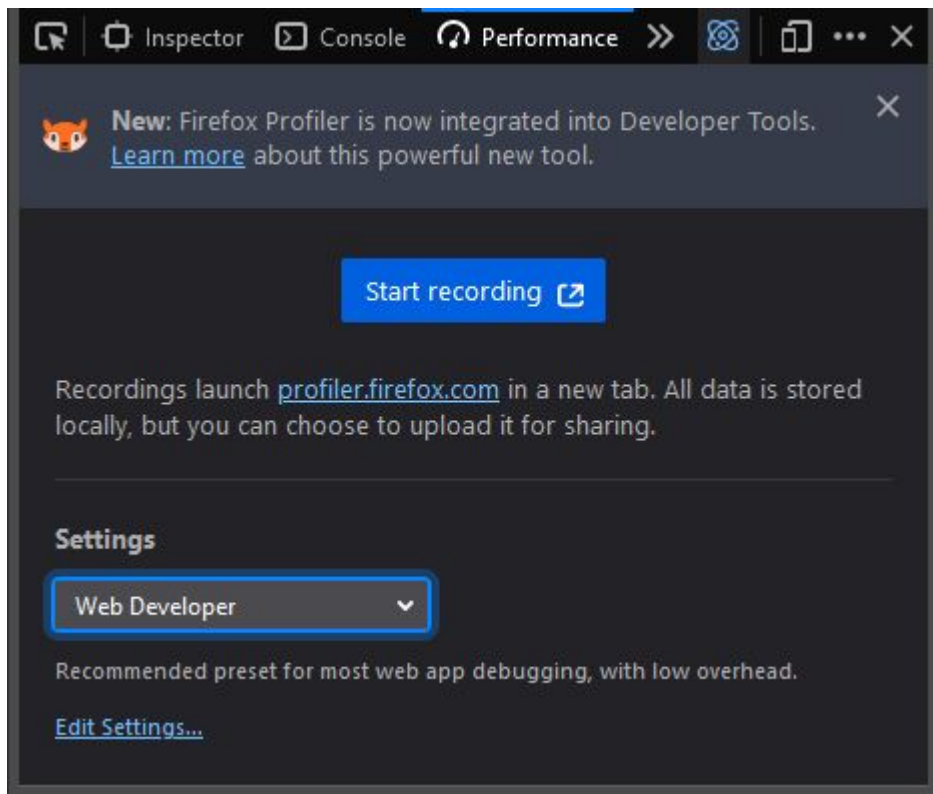
Categories Running sample count

- JavaScript 54% 44
- Layout 23% 19
- GC / CC 9.8% 8
- DOM 7.3% 6

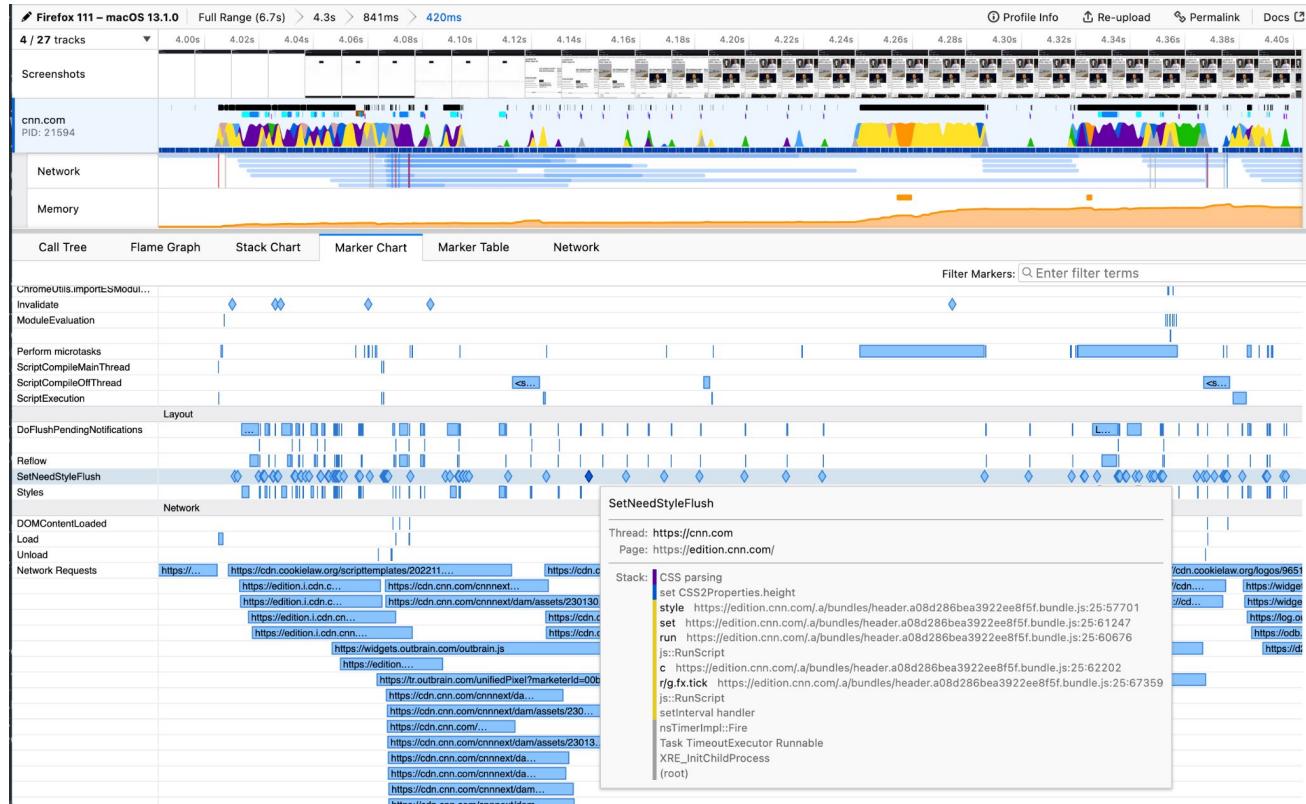
```
js/src/vm/Interpreter.h
Total Self
111
112 inline bool Call(JSContext* cx, HandleValue fval, HandleValue thisv,
113                 HandleValue arg0, MutableHandleValue rval) {
114     FixedInvokeArgs<1> args(cx);
115     args[0].set(arg0);
84 116     return Call(cx, fval, thisv, args, rval);
117 }
118
119 inline bool Call(JSContext* cx, HandleValue fval, JSObject* thisObj,
120                 HandleValue arg0, MutableHandleValue rval) {
121     RootedValue thisv(cx, ObjectOrNullValue(thisObj));
122     FixedInvokeArgs<1> args(cx);
123     args[0].set(arg0);
124     return Call(cx, fval, thisv, args, rval);
```



New DevTools performance panel

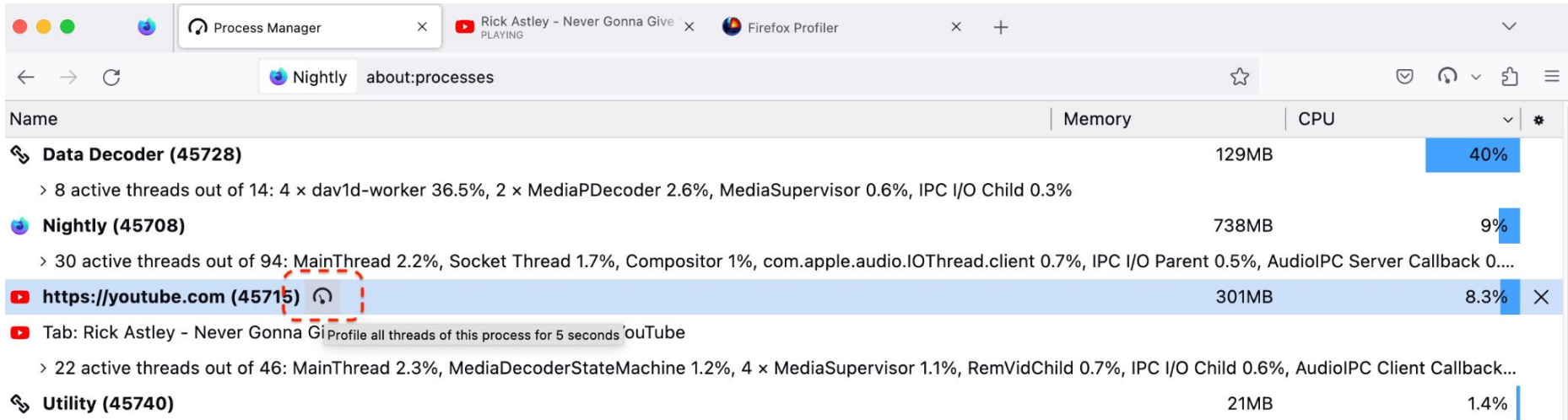


Marker Chart improvements











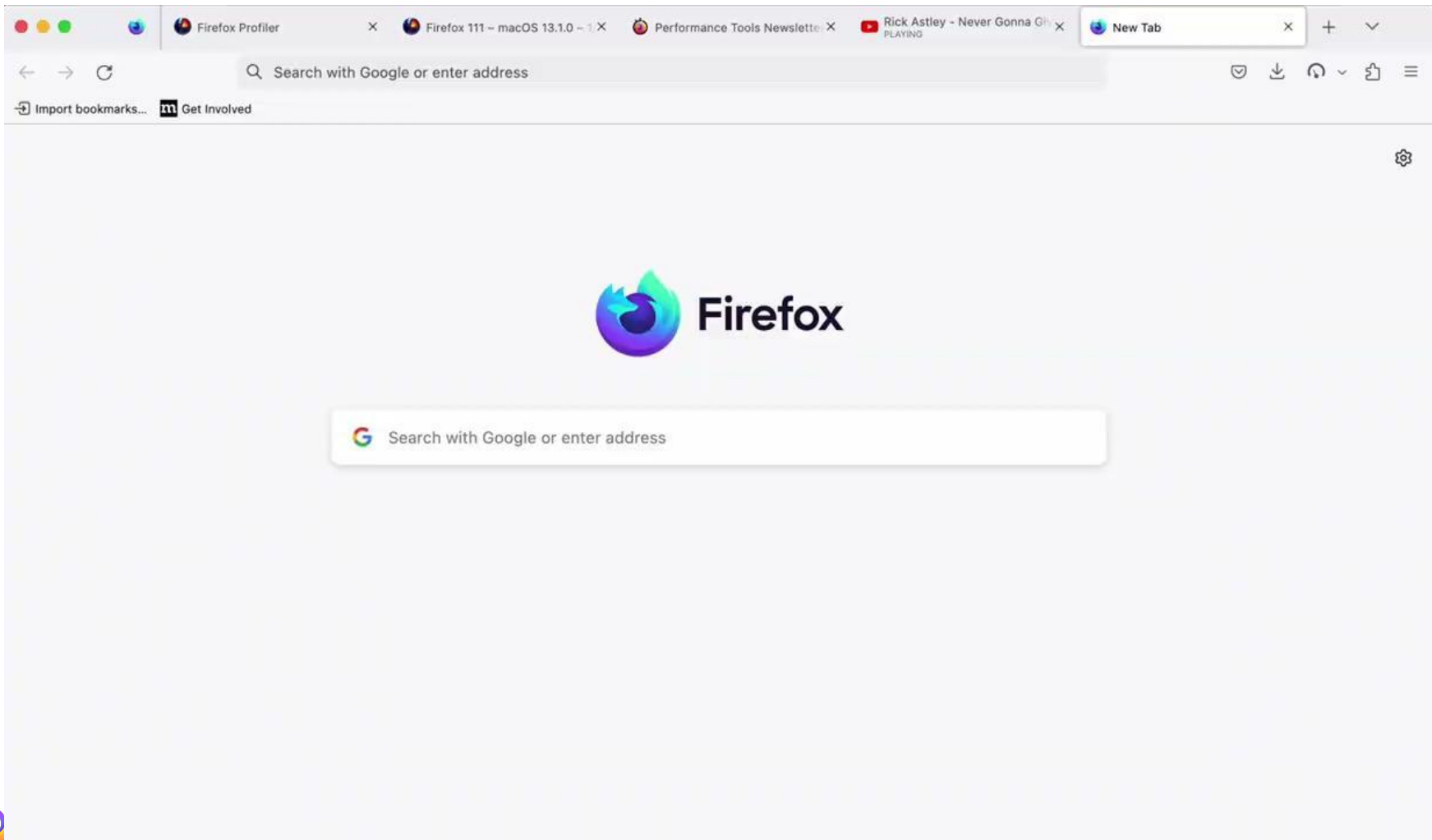
One-click profiling on Task Manager (a.k.a. about:processes)



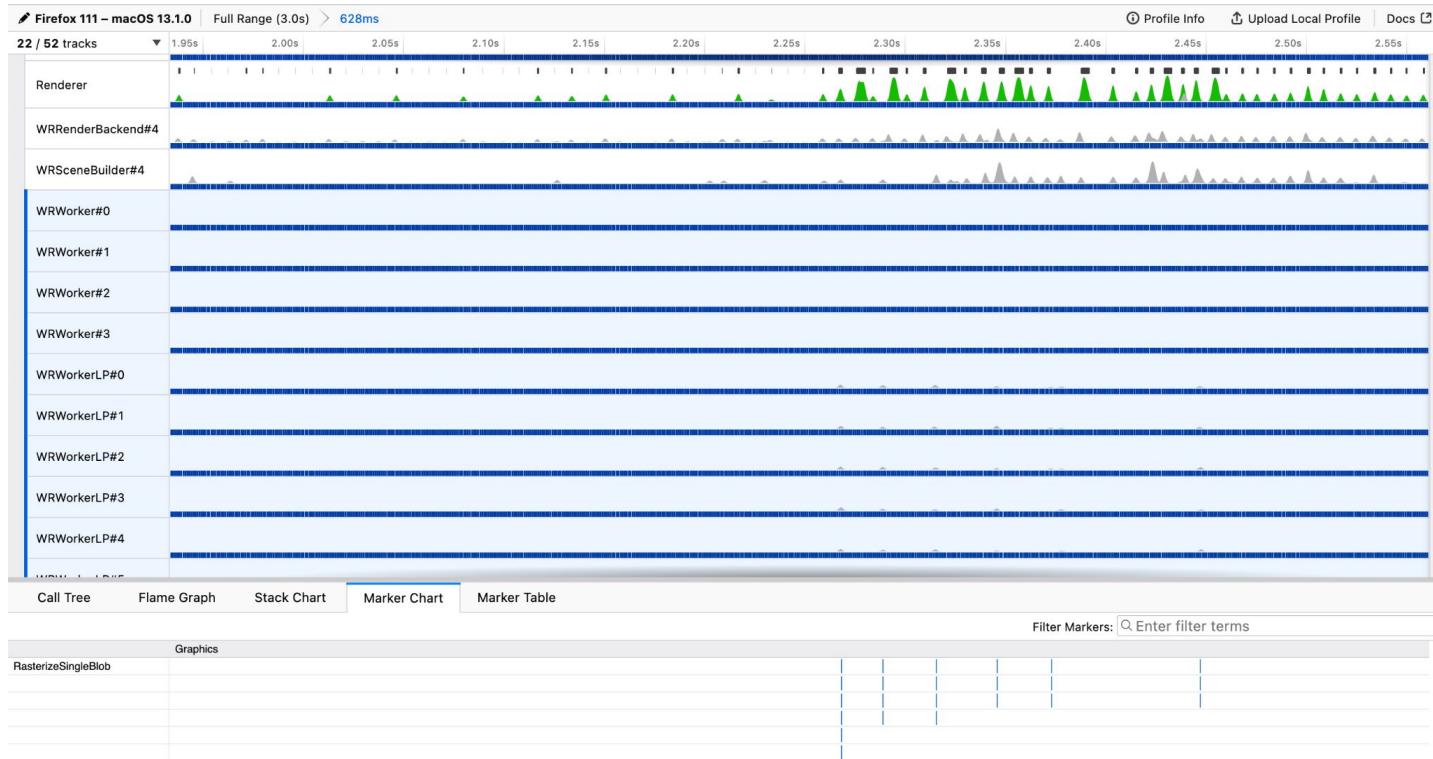
The screenshot shows the Firefox Profiler interface with the 'about:processes' page open. The browser tabs include 'Process Manager', 'Rick Astley - Never Gonna Give', and 'Firefox Profiler'. The address bar shows 'Nightly about:processes'. The main content area displays a table of processes with columns for Name, Memory, and CPU. The 'https://youtube.com (45715)' process is highlighted in blue, and its CPU usage bar is also blue, indicating it is the selected process for profiling. A red dashed box highlights the 'Profile' icon (a circular arrow) next to the process name.

Name	Memory	CPU
 Data Decoder (45728) > 8 active threads out of 14: 4 x dav1d-worker 36.5%, 2 x MediaPDecoder 2.6%, MediaSupervisor 0.6%, IPC I/O Child 0.3%	129MB	40%
 Nightly (45708) > 30 active threads out of 94: MainThread 2.2%, Socket Thread 1.7%, Compositor 1%, com.apple.audio.IOThread.client 0.7%, IPC I/O Parent 0.5%, AudioIPC Server Callback 0...	738MB	9%
 https://youtube.com (45715) 	301MB	8.3%
 Tab: Rick Astley - Never Gonna Give a Minute > 22 active threads out of 46: MainThread 2.3%, MediaDecoderStateMachine 1.2%, 4 x MediaSupervisor 1.1%, RemVidChild 0.7%, IPC I/O Child 0.6%, AudioIPC Client Callback...		
 Utility (45740)	21MB	1.4%





Select multiple tracks



Use CTRL+Click or Shift+Click to select multiple tracks quickly.



Tracks selection list

The screenshot displays the Performance panel in Chrome DevTools for a Firefox 111 instance on macOS 13.1.0. The top navigation bar shows the full range (6.7s) and a zoomed-in view of 420ms. A search bar for filter terms is present. The tracks selection list on the left includes:

- Show all tracks
- Screenshots
- Parent Process (21584) ✓
 - Network ✓
 - Memory ✓
 - CanvasRenderer
 - Compositor ✓
 - DOM Worker ✓
 - DOM Worker ✓
 - DOM Worker ✓
 - DOM Worker ✓
 - Renderer ✓
- Web Content (1/3) (21604) ✓
 - Memory
- Socket Process (21588) ✓
 - Memory
- Web Content (2/3) (21596) ✓
 - Memory
- WebExtensions (21589) ✓
 - Memory
- Web Content (3/3) (21595) ✓
 - Memory
- Privileged Content (21592) ✓
 - Memory
 - DOM Worker
- cnn.com (21594) ✓
 - DOM Worker
- 100%*work 1.676 ✓

The right side of the panel shows a call stack for the selected track. The stack includes:

- (root)
- pthread_start libsystem_pthread.dylib
- _pt_root nsprpub/pr/src/pthreads/ptthread.c
- nsThread::ThreadFunc(void*) xpcom/threads/nsThread.c
- MessageLoop::Run() ipc/chromium/src/base/message_
- MessageLoop::RunHandler() ipc/chromium/src/ba
- MessageLoop::RunInternal() ipc/chromium/src/t
- mozilla::ipc::MessagePumpForNonMainThreads::R
- NS_ProcessNextEvent(nsIThread*, bool) xpc



New transform #1: Focus on Category

Complete "https://cnn.com" > Drop: __psynch_cwait

- Merge function
- Merge node only
- Focus on function
- Focus on subtree only
- Focus on category JavaScript**
- Focusing on the nodes that belong to the same category as the selected node, thereby merging all nodes that belong to another category.
- Look up the function name on Searchfox
- Copy function name
- Copy stack

Thanks to our contributor **Johannes Bechberger (SAP)**!



New transform #2: Collapse Indirect Recursion

The screenshot displays the Performance tab in Chrome DevTools, showing a flame graph and call tree for a page load on cnn.com. The flame graph at the top shows various colored bars representing different functions and their execution time. Below it, the call tree is visible, showing a sequence of function calls. A context menu is open over a function node, listing several actions: Merge function, Merge node only, Focus on function, Focus on subtree only, Focus on category JavaScript, Collapse function, Collapse indirect recursion (highlighted), Drop samples with this function, Look up the function name on Searchfox, Copy function name, and Copy stack. The 'Collapse indirect recursion' option is highlighted in blue, and a tooltip explains that collapsing indirect recursion removes calls that repeatedly recurse into the same function, even with intermediate functions on the stack.

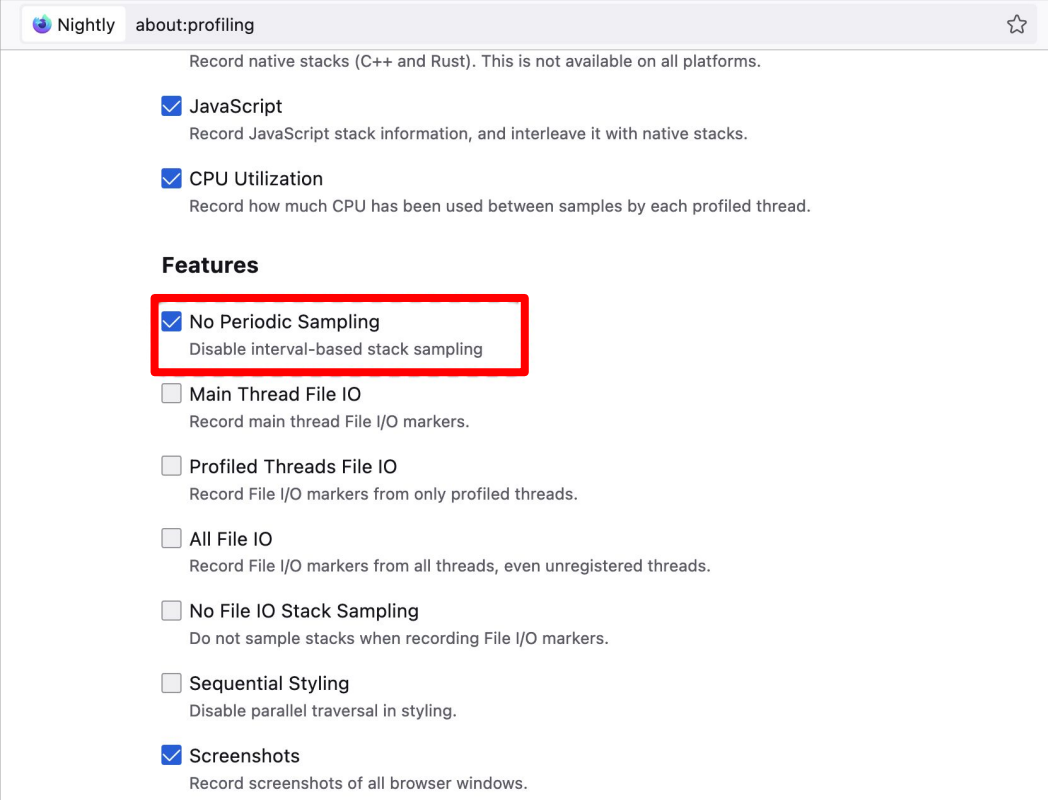
Thanks to our contributor **Simmo Saan!**



Locale picker



No periodic stack sampling feature



Nightly about:profiling

Record native stacks (C++ and Rust). This is not available on all platforms.

- JavaScript
Record JavaScript stack information, and interleave it with native stacks.
- CPU Utilization
Record how much CPU has been used between samples by each profiled thread.

Features

- No Periodic Sampling
Disable interval-based stack sampling
- Main Thread File IO
Record main thread File I/O markers.
- Profiled Threads File IO
Record File I/O markers from only profiled threads.
- All File IO
Record File I/O markers from all threads, even unregistered threads.
- No File IO Stack Sampling
Do not sample stacks when recording File I/O markers.
- Sequential Styling
Disable parallel traversal in styling.
- Screenshots
Record screenshots of all browser windows.



Documentation

Firefox Profiler

User Guide

Getting Started

Access the Firefox Profiler controls

Capture a profile

Share a profile

Delete an uploaded profile

Name a profile

UI Tour: The Timeline

UI Tour: The Panels

Profiler Fundamentals

Stack samples and call trees

Filtering call trees

Profiling Firefox for Android

Remote Profiling

Profiling directly on the device

Memory Allocations

Advanced Topics

Profiling Firefox Startup & Shutdown

Perf profiling on Linux

Getting Started

This page is an overview of the general process of capturing a profile.

Access the Firefox Profiler controls

The Firefox Profiler can be used with 2 entry points: either from the popup, or from the devtools panel.

We recommend using the popup so that the overhead caused by the other devtools is avoided.

Enable the popup

The simplest way to enable the popup is going to <https://profiler.firefox.com> and clicking the big button `Enable Firefox Profiler Menu Button` in the page.

<https://profiler.firefox.com/docs/>



Thank you!

Firefox Profiler docs:

<https://profiler.firefox.com/docs/>

Slides:

<https://share.firefox.dev/profiler-whatsnew-fosdem-2023>

Matrix channel:

<https://chat.mozilla.org/#/room/#profiler:mozilla.org>

