JS Profiling: Updates and State Extension

acomminos@fb.com
cpescheloche@fb.com

Recap

API overview

- A web-exposed sampling profiler for client JS execution
- Provide insight into client performance characteristics on real users' devices
- Shipped in Chrome 94

- github.com/WICG/js-self-profiling

```
const profiler = new Profiler({ sampleInterval: 10, maxBufferSize: 10000 });
```

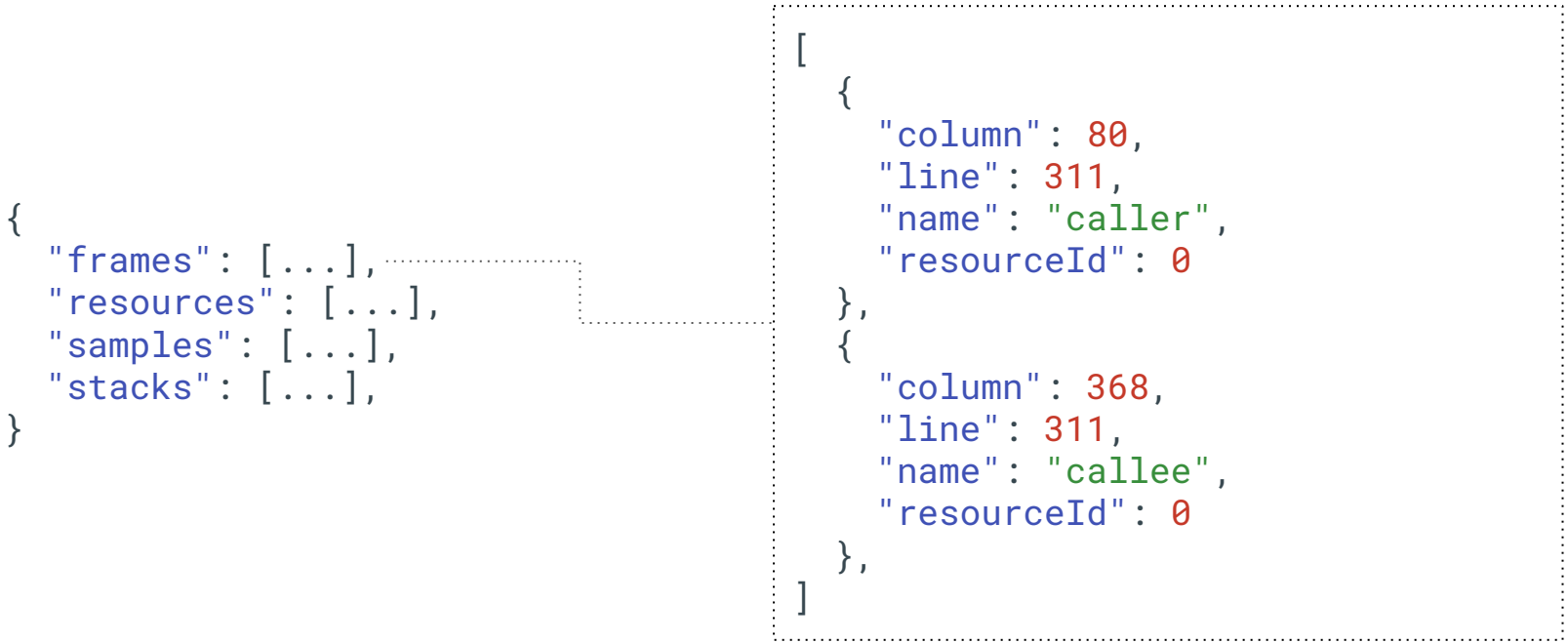
Starting a profiler

```
const profiler = new Profiler({ sampleInterval: 10, maxBufferSize: 10000 });

window.addEventListener('load', async () => {
  const trace = await profiler.stop();
  const traceJson = JSON.stringify(trace);
  sendTrace(traceJson);
});
```

Sending trace on load

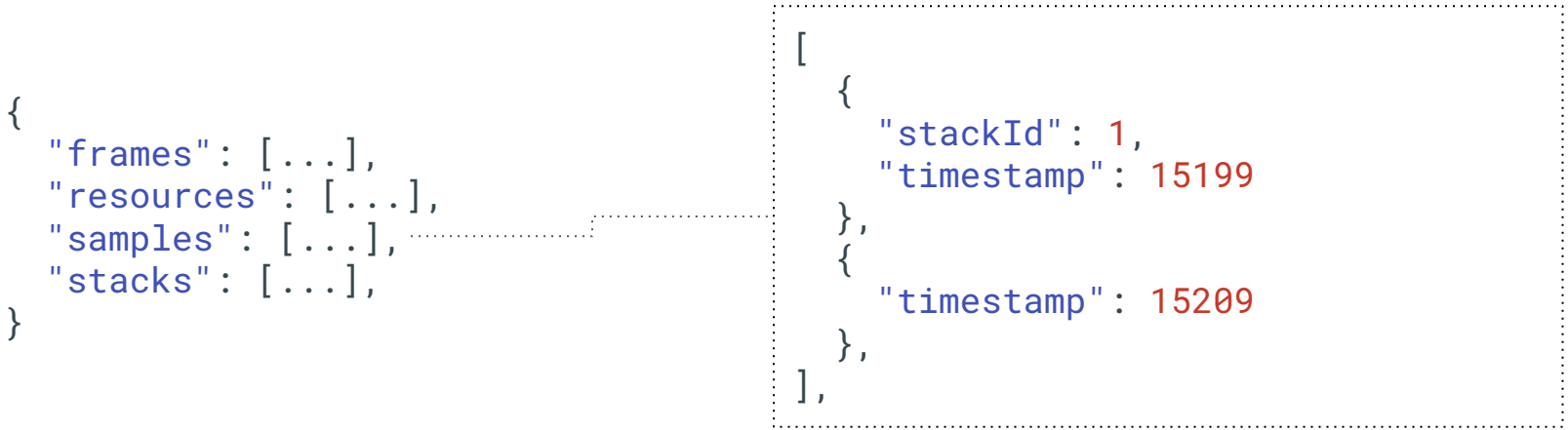
```
const profiler = new Profiler({ sampleInterval: 10, maxBufferSize: 10000 });

window.addEventListener('load', async () => {
  const trace = await profiler.stop();
  const traceJson = JSON.stringify({
    entries: performance.getEntries(),
    trace,
  });
  sendTrace(traceJson);
});
```

Including data from the performance timeline

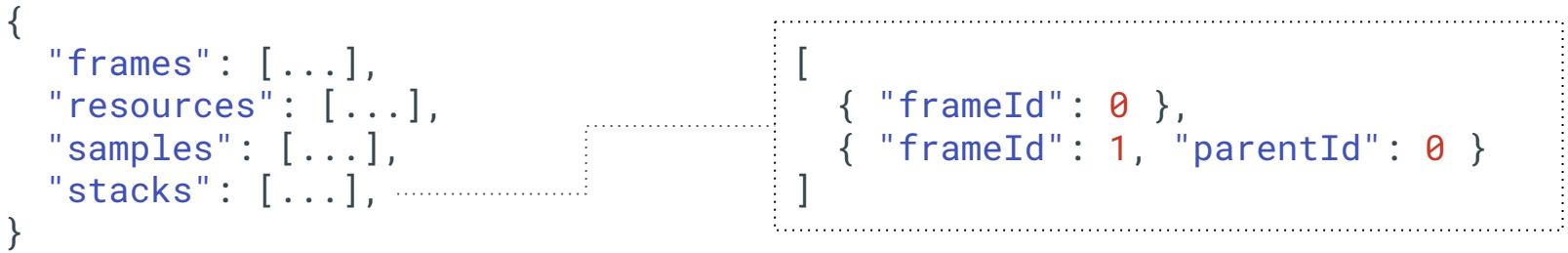


Trace format: frames

Trace format: resources



Trace format: samples



Trace format: stacks

What's working well?

- Initial data suggests **enabling profiling slows load time by <1% (p=0.05)** at FB
  - Strong evidence that sampling profiling can be implemented with minimal overhead
- API has provided a drop-in solution for FB app client perf analysis
- Strong adoption from other industry partners (Microsoft)

What could be better?

- Non-JS execution is hard to identify in traces
  - Currently, top-level UA work is indistinguishable from idle execution
  - GC activity adds to the noise of long traces
  - Client code that causes asynchronous rendering work isn't measurable

Representing non-JS execution

Introducing state markers

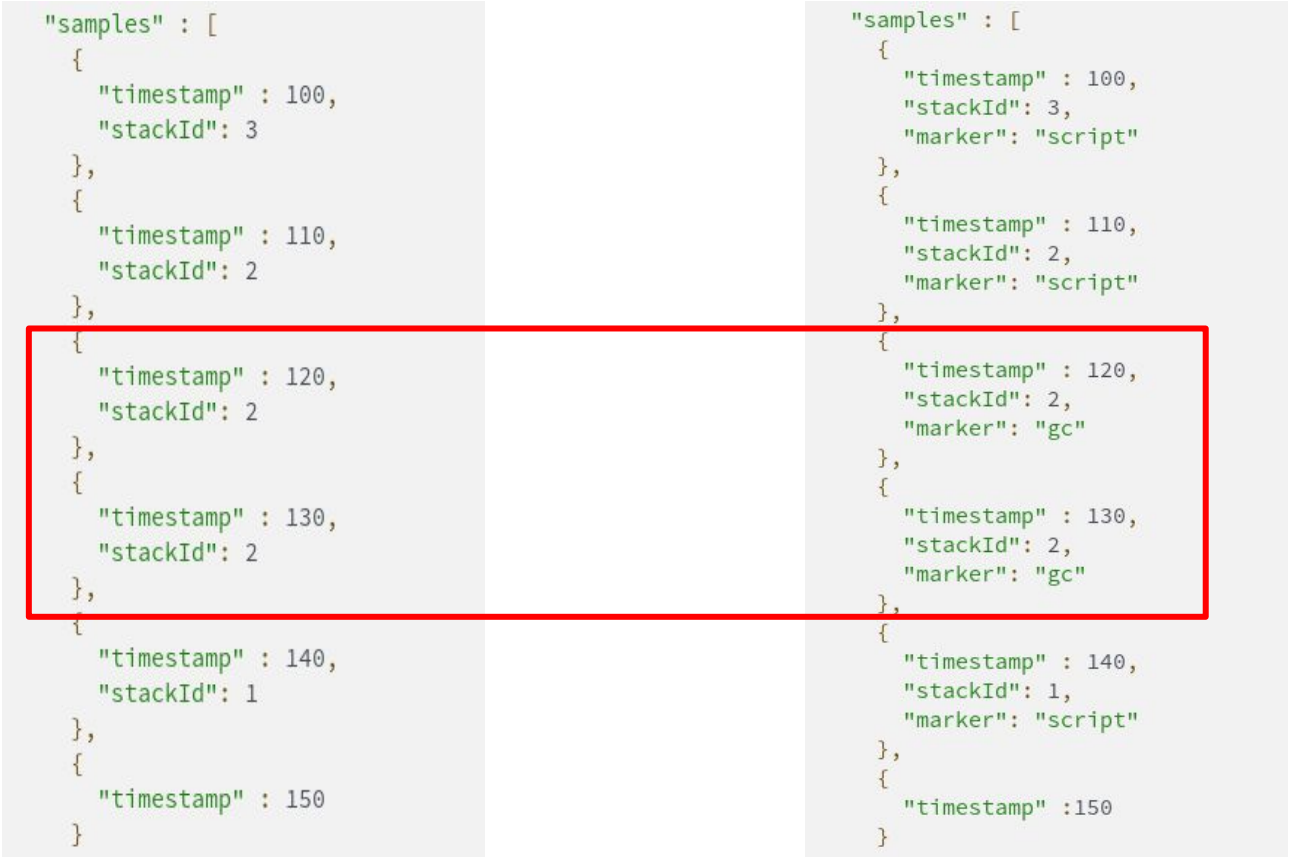
- Tags a sample with a string representing top level UA work category
- Similar representation to traces visualized through devtool profilers

Marker candidates

- Need to be generic and interoperable
- Script related:
  - script: js execution, optional ?
  - parse: HTML? JS?
  - gc
- Rendering related:
  - paint: update the rendering part of the event loop or limited to actual paint
  - style
  - layout

API Modification

```
enum ProfilerMarker { "script", "gc", "parse", "paint", "other" };

dictionary ProfilerSample {
  required DOMHighResTimeStamp timestamp;
  unsigned long long stackId;
  ProfilerMarker? marker;
};
```



Example trace GC

Security and Privacy concerns

- Profiles **must not** expose work done on a cross-origin document
  - Top level UA work may only appear in a trace if the **responsible document** for the work is same-origin with the associated Profiler
- New information exposed, need to limit granularity of marker types
  - Need to avoid introducing new side channels
  - Require cross-origin isolation for markers?

Open questions

Markers

- Interest in breaking down paint marker into:
  - Style
  - Layout
  - Paint
- Require cross origin isolation for events like GC

Correct place?

- Is JS self profiling the best place for this information?
  - Performance-timeline could be a candidate

Links

- Explainer: https://github.com/WICG/js-self-profiling/pull/55

Appendix

```
[Exposed=Window]
interface Profiler : EventTarget {
  readonly attribute DOMHighResTimeStamp sampleInterval;
  readonly attribute boolean stopped;

  constructor(ProfilerInitOptions options);
  Promise<ProfilerTrace> stop();
};
```

IDL: Profiler