

V8 Options

Updated: 2023-12-05

SSE3=1 SSSE3=1 SSE4_1=1 SSE4_2=1 SAHF=1 AVX=1 AVX2=1 FMA3=1 BMI1=1 BMI2=1 LZCNT=1 POPCNT=1 ATOM=0

The following syntax for options is accepted (both '-' and '--' are ok):

```
--flag      (bool flags only)
--no-flag    (bool flags only)
--flag=value (non-bool flags only, no spaces around '=')
--flag value (non-bool flags only)
--          (captures all remaining args in JavaScript)
```

Options:

--experimental (Indicates that V8 is running with experimental features enabled. This flag is typically not set explicitly but instead enabled as an implication of other flags which enable experimental features.)

type: bool default: --no-experimental

--abort-on-contradictory-flags (Disallow flags or implications overriding each other.)

type: bool default: --no-abort-on-contradictory-flags

--exit-on-contradictory-flags (Exit with return code 0 on contradictory flags.)

type: bool default: --no-exit-on-contradictory-flags

--allow-overwriting-for-next-flag (temporary disable flag contradiction to allow overwriting just the next flag)

type: bool default: --no-allow-overwriting-for-next-flag

--use-strict (enforce strict mode)

type: bool default: --no-use-strict

--trace-temporal (trace temporal code)

type: bool default: --no-trace-temporal

--harmony (enable all completed harmony features)

type: bool default: --no-harmony

--harmony-shipping (enable all shipped harmony features)

type: bool default: --harmony-shipping

--harmony-weak-refs-with-cleanup-some (enable "harmony weak references with FinalizationRegistry.prototype.cleanupSome" (in progress / experimental))

type: bool default: --no-harmony-weak-refs-with-cleanup-some

--harmony-temporal (enable "Temporal" (in progress / experimental))

type: bool default: --no-harmony-temporal

--harmony-shadow-realm (enable "harmony ShadowRealm" (in progress / experimental))

type: bool default: --no-harmony-shadow-realm

--harmony-struct (enable "harmony structs, shared structs, and shared arrays" (in progress / experimental))

type: bool default: --no-harmony-struct

--harmony-array-from-async (enable "harmony Array.fromAsync" (in progress / experimental))

type: bool default: --no-harmony-array-from-async

--harmony-iterator-helpers (enable "JavaScript iterator helpers" (in progress / experimental))

type: bool default: --no-harmony-iterator-helpers

--harmony-intl-best-fit-matcher (enable "Intl BestFitMatcher" (in progress / experimental))

type: bool default: --no-harmony-intl-best-fit-matcher

--harmony-intl-duration-format (enable "Intl DurationFormat API" (in progress / experimental))

type: bool default: --no-harmony-intl-duration-format

--harmony-rab-gsab-transfer (enable "harmony ArrayBuffer.transfer")

type: bool default: --no-harmony-rab-gsab-transfer

--harmony-array-grouping (enable "harmony array grouping")

type: bool default: --no-harmony-array-grouping

--harmony-json-parse-with-source (enable "harmony json parse with source")

type: bool default: --no-harmony-json-parse-with-source

--harmony-sharedarraybuffer (enable "harmony sharedarraybuffer")

type: bool default: --harmony-sharedarraybuffer

--harmony-atomics (enable "harmony atomics")

type: bool default: --harmony-atomics

--harmony-import-assertions (enable "harmony import assertions")

type: bool default: --harmony-import-assertions

--harmony-symbol-as-weakmap-key (enable "harmony symbols as weakmap keys")

type: bool default: --harmony-symbol-as-weakmap-key

--harmony-change-array-by-copy (enable "harmony change-Array-by-copy")

type: bool default: --harmony-change-array-by-copy

--harmony-string-is-well-formed (enable "harmony String#{is,to}WellFormed")

type: bool default: --harmony-string-is-well-formed

--harmony-rab-gsab (enable "harmony ResizableArrayBuffer / GrowableSharedArrayBuffer")

type: bool default: --harmony-rab-gsab

--harmony-regexp-unicode-sets (enable "harmony RegExp Unicode Sets")

type: bool default: --harmony-regexp-unicode-sets

```
--harmony-intl-number-format-v3 (enable "Intl.NumberFormat v3")
    type: bool  default: --harmony-intl-number-format-v3
--builtin-subclassing (subclassing support in built-in methods)
    type: bool  default: --builtin-subclassing
--enable-sharedarraybuffer-per-context (enable the SharedArrayBuffer constructor per context)
    type: bool  default: --no-enable-sharedarraybuffer-per-context
--icu-timezone-data (get information about timezones from ICU)
    type: bool  default: --icu-timezone-data
--stress-snapshot (disables sharing of the read-only heap for testing)
    type: bool  default: --no-stress-snapshot
--lite-mode (enables trade-off of performance for memory savings)
    type: bool  default: --no-lite-mode
--enable-third-party-heap (Use third-party heap)
    type: bool  default: --no-enable-third-party-heap
--enable-allocation-folding (Use allocation folding globally)
    type: bool  default: --enable-allocation-folding
--disable-write-barriers (disable write barriers when GC is non-incremental and heap contains single
generation.)
    type: bool  default: --no-disable-write-barriers
--enable-unconditional-write-barriers (always use full write barriers)
    type: bool  default: --no-enable-unconditional-write-barriers
--single-generation (allocate all objects from young generation to old generation)
    type: bool  default: --no-single-generation
--conservative-stack-scanning (use conservative stack scanning)
    type: bool  default: --no-conservative-stack-scanning
--future (Implies all staged features that we want to ship in the not-too-far future)
    type: bool  default: --no-future
--force-emit-interrupt-budget-checks (force emit tier-up logic from all non-turbofan code, even if it is the
top enabled tier)
    type: bool  default: --no-force-emit-interrupt-budget-checks
--maglev (enable the maglev optimizing compiler)
    type: bool  default: --no-maglev
--maglev-future (enable maglev features that we want to ship in the not-too-far future)
    type: bool  default: --no-maglev-future
--maglev-inlining (enable inlining in the maglev optimizing compiler)
    type: bool  default: --no-maglev-inlining
--maglev-untagged-phis (enable phi untagging in the maglev optimizing compiler)
    type: bool  default: --no-maglev-untagged-phis
--stress-maglev (trigger maglev compilation earlier)
    type: bool  default: --no-stress-maglev
--optimize-on-next-call-optimizes-to-maglev (make OptimizeFunctionOnNextCall optimize to maglev instead of
turbofan)
    type: bool  default: --no-optimize-on-next-call-optimizes-to-maglev
--maglev-filter (optimization filter for the maglev compiler)
    type: string default: --maglev-filter="*"
--maglev-assert (insert extra assertion in maglev code)
    type: bool  default: --no-maglev-assert
--maglev-assert-stack-size (insert stack size checks before every IR node)
    type: bool  default: --maglev-assert-stack-size
--maglev-break-on-entry (insert an int3 on maglev entries)
    type: bool  default: --no-maglev-break-on-entry
--print-maglev-graph (print maglev graph)
    type: bool  default: --no-print-maglev-graph
--print-maglev-deopt-verbose (print verbose deopt info)
    type: bool  default: --no-print-maglev-deopt-verbose
--print-maglev-code (print maglev code)
    type: bool  default: --no-print-maglev-code
--trace-maglev-graph-building (trace maglev graph building)
    type: bool  default: --no-trace-maglev-graph-building
--trace-maglev-regalloc (trace maglev register allocation)
    type: bool  default: --no-trace-maglev-regalloc
--trace-maglev-inlining (trace maglev inlining)
    type: bool  default: --no-trace-maglev-inlining
--trace-maglev-inlining-verbose (trace maglev inlining (verbose))
    type: bool  default: --no-trace-maglev-inlining-verbose
--maglev-function-context-specialization (enable function context specialization in maglev)
    type: bool  default: --maglev-function-context-specialization
--dict-property-const-tracking (Use const tracking on dictionary properties)
    type: bool  default: --no-dict-property-const-tracking
```

```
--max-opt (Set the maximal optimisation tier: > 3 == any, 0 == ignition/interpreter, 1 == sparkplug/baseline, 2 == maglev, 3 == turbofan)
    type: uint  default: --max-opt=999
--wasm-trace-native (Select which native code sequence to use for wasm trace instruction: default or cpuid)
    type: string default: --wasm-trace-native=""
--jitless (Disable runtime allocation of executable memory.)
    type: bool  default: --no-jitless
--assert-types (generate runtime type assertions to test the typer)
    type: bool  default: --no-assert-types
--turboshaft-assert-types (generate runtime type assertions to test the turboshaft type system)
    type: bool  default: --no-turboshaft-assert-types
--verify-simplified-lowering (verify graph generated by simplified lowering)
    type: bool  default: --no-verify-simplified-lowering
--trace-compilation-dependencies (trace code dependencies)
    type: bool  default: --no-trace-compilation-dependencies
--allocation-site-tracking (Enable allocation site tracking)
    type: bool  default: --allocation-site-tracking
--allocation-site-pretenuing (pretenuing with allocation sites)
    type: bool  default: --allocation-site-pretenuing
--page-promotion (promote pages based on utilization)
    type: bool  default: --page-promotion
--page-promotion-threshold (min percentage of live bytes on a page to enable fast evacuation)
    type: int   default: --page-promotion-threshold=70
--trace-pretenuing (trace pretenuing decisions of HAllocate instructions)
    type: bool  default: --no-trace-pretenuing
--trace-pretenuing-statistics (trace allocation site pretenuing statistics)
    type: bool  default: --no-trace-pretenuing-statistics
--track-field-types (track field types)
    type: bool  default: --track-field-types
--trace-block-coverage (trace collected block coverage information)
    type: bool  default: --no-trace-block-coverage
--trace-protector-invalidation (trace protector cell invalidations)
    type: bool  default: --no-trace-protector-invalidation
--feedback-normalization (feed back normalization to constructors)
    type: bool  default: --no-feedback-normalization
--internalize-on-the-fly (internalize string keys for generic keyed ICs on the fly)
    type: bool  default: --internalize-on-the-fly
--enable-sealed-frozen-elements-kind (Enable sealed, frozen elements kind)
    type: bool  default: --enable-sealed-frozen-elements-kind
--unbox-double-arrays (automatically unbox arrays of doubles)
    type: bool  default: --unbox-double-arrays
--string-slices (use string slices)
    type: bool  default: --string-slices
--interrupt-budget-for-feedback-allocation (The fixed interrupt budget (in bytecode size) for allocating feedback vectors)
    type: int   default: --interrupt-budget-for-feedback-allocation=940
--interrupt-budget-factor-for-feedback-allocation (The interrupt budget factor (applied to bytecode size) for allocating feedback vectors, used when bytecode size is known)
    type: int   default: --interrupt-budget-factor-for-feedback-allocation=8
--invocation-count-for-maglev (interrupt budget which should be used for the profiler counter)
    type: int   default: --invocation-count-for-maglev=100
--interrupt-budget (interrupt budget which should be used for the profiler counter)
    type: int   default: --interrupt-budget=67584
--ticks-before-optimization (the number of times we have to go through the interrupt budget before considering this function for optimization)
    type: int   default: --ticks-before-optimization=3
--bytecode-size-allowance-per-tick (increases the number of ticks required for optimization by bytecode.length/X)
    type: int   default: --bytecode-size-allowance-per-tick=150
--invocation-count-for-osr (number of invocations we want to see after requesting previous tier up to increase the OSR urgency)
    type: int   default: --invocation-count-for-osr=500
--max-bytecode-size-for-early-opt (Maximum bytecode length for a function to be optimized on the first tick)
    type: int   default: --max-bytecode-size-for-early-opt=81
--global-ic-updated-flag (Track, globally, whether any IC changed, and use this in tierup heuristics.)
    type: bool  default: --no-global-ic-updated-flag
--minimum-invocations-after-ic-update (How long to minimally wait after IC update before tier up)
    type: int   default: --minimum-invocations-after-ic-update=500
--reset-interrupt-on-ic-update (On IC change, reset the interrupt budget for just that function.)
```

```
    type: bool  default: --reset-interrupt-on-ic-update
--reset-ticks-on-ic-update (On IC change, reset the ticks for just that function.)
    type: bool  default: --reset-ticks-on-ic-update
--maglev-increase-budget-forward-jump (Increase interrupt budget on forward jumps in maglev code)
    type: bool  default: --no-maglev-increase-budget-forward-jump
--use-ic (use inline caching)
    type: bool  default: --use-ic
--lazy-feedback-allocation (Allocate feedback vectors lazily)
    type: bool  default: --lazy-feedback-allocation
--ignition-elide-noneffectful-bytecodes (elide bytecodes which won't have any external effect)
    type: bool  default: --ignition-elide-noneffectful-bytecodes
--ignition-reo (use ignition register equivalence optimizer)
    type: bool  default: --ignition-reo
--ignition-filter-expression-positions (filter expression positions before the bytecode pipeline)
    type: bool  default: --ignition-filter-expression-positions
--ignition-share-named-property-feedback (share feedback slots when loading the same named property from the
same object)
    type: bool  default: --ignition-share-named-property-feedback
--print-bytecode (print bytecode generated by ignition interpreter)
    type: bool  default: --no-print-bytecode
--enable-lazy-source-positions (skip generating source positions during initial compile but regenerate when
actually required)
    type: bool  default: --enable-lazy-source-positions
--stress-lazy-source-positions (collect lazy source positions immediately after lazy compile)
    type: bool  default: --no-stress-lazy-source-positions
--print-bytecode-filter (filter for selecting which functions to print bytecode)
    type: string default: --print-bytecode-filter="*"
--omit-default-ctors (omit calling default ctors in bytecode)
    type: bool  default: --omit-default-ctors
--trace-ignition-codegen (trace the codegen of ignition interpreter bytecode handlers)
    type: bool  default: --no-trace-ignition-codegen
--trace-ignition-dispatches-output-file (write the bytecode handler dispatch table to the specified file (d8
only) (requires building with v8_enable_ignition_dispatch_counting))
    type: string default: --trace-ignition-dispatches-output-file=""
--trace-track-allocation-sites (trace the tracking of allocation sites)
    type: bool  default: --no-trace-track-allocation-sites
--trace-migration (trace object migration)
    type: bool  default: --no-trace-migration
--trace-generalization (trace map generalization)
    type: bool  default: --no-trace-generalization
--sparkplug (enable Sparkplug baseline compiler)
    type: bool  default: --sparkplug
--always-sparkplug (directly tier up to Sparkplug code)
    type: bool  default: --no-always-sparkplug
--baseline-batch-compilation (batch compile Sparkplug code)
    type: bool  default: --baseline-batch-compilation
--concurrent-sparkplug (compile Sparkplug code in a background thread)
    type: bool  default: --no-concurrent-sparkplug
--concurrent-sparkplug-max-threads (max number of threads that concurrent Sparkplug can use (0 for unbounded))
    type: uint  default: --concurrent-sparkplug-max-threads=1
--concurrent-sparkplug-high-priority-threads (use high priority compiler threads for concurrent Sparkplug)
    type: bool  default: --no-concurrent-sparkplug-high-priority-threads
--sparkplug-filter (filter for Sparkplug baseline compiler)
    type: string default: --sparkplug-filter="*"
--sparkplug-needs-short-builtins (only enable Sparkplug baseline compiler when --short-builtin-calls are also
enabled)
    type: bool  default: --no-sparkplug-needs-short-builtins
--baseline-batch-compilation-threshold (the estimated instruction size of a batch to trigger compilation)
    type: int   default: --baseline-batch-compilation-threshold=4096
--trace-baseline (trace baseline compilation)
    type: bool  default: --no-trace-baseline
--trace-baseline-batch-compilation (trace baseline batch compilation)
    type: bool  default: --no-trace-baseline-batch-compilation
--trace-baseline-concurrent-compilation (trace baseline concurrent compilation)
    type: bool  default: --no-trace-baseline-concurrent-compilation
--shared-string-table (internalize strings into shared table)
    type: bool  default: --no-shared-string-table
--always-use-string-forwarding-table (use string forwarding table instead of thin strings for all strings)
    type: bool  default: --no-always-use-string-forwarding-table
```

```
--transition-strings-during-gc-with-stack (Transition strings during a full GC with stack)
    type: bool  default: --no-transition-strings-during-gc-with-stack
--initial-shared-heap-size (initial size of the shared heap (in Mbytes); other heap size flags (e.g.
initial_heap_size) take precedence)
    type: size_t  default: --initial-shared-heap-size=0
--max-shared-heap-size (max size of the shared heap (in Mbytes); other heap size flags (e.g.
max_shared_heap_size) take precedence)
    type: size_t  default: --max-shared-heap-size=0
--write-code-using-rwx (flip permissions to rwx to write page instead of rw)
    type: bool  default: --write-code-using-rwx
--concurrent-recompilation (optimizing hot functions asynchronously on a separate thread)
    type: bool  default: --concurrent-recompilation
--trace-concurrent-recompilation (track concurrent recompilation)
    type: bool  default: --no-trace-concurrent-recompilation
--concurrent-recompilation-queue-length (the length of the concurrent compilation queue)
    type: int  default: --concurrent-recompilation-queue-length=8
--concurrent-recompilation-delay (artificial compilation delay in ms)
    type: int  default: --concurrent-recompilation-delay=0
--stress-concurrent-inlining (create additional concurrent optimization jobs but throw away result)
    type: bool  default: --no-stress-concurrent-inlining
--maglev-overwrite-budget (whether maglev resets the interrupt budget)
    type: bool  default: --no-maglev-overwrite-budget
--stress-concurrent-inlining-attach-code (create additional concurrent optimization jobs)
    type: bool  default: --no-stress-concurrent-inlining-attach-code
--max-serializer-nesting (maximum levels for nesting child serializers)
    type: int  default: --max-serializer-nesting=25
--trace-heap-broker-verbose (trace the heap broker verbosely (all reports))
    type: bool  default: --no-trace-heap-broker-verbose
--trace-heap-broker-memory (trace the heap broker memory (refs analysis and zone numbers))
    type: bool  default: --no-trace-heap-broker-memory
--trace-heap-broker (trace the heap broker (reports on missing data only))
    type: bool  default: --no-trace-heap-broker
--stress-runs (number of stress runs)
    type: int  default: --stress-runs=0
--deopt-every-n-times (deoptimize every n times a deopt point is passed)
    type: int  default: --deopt-every-n-times=0
--print-deopt-stress (print number of possible deopt points)
    type: bool  default: --no-print-deopt-stress
--turbofan (use the Turbofan optimizing compiler)
    type: bool  default: --turbofan
--opt (alias for --turbofan)
    type: bool  default: --opt
--turbo-sp-frame-access (use stack pointer-relative access to frame wherever possible)
    type: bool  default: --no-turbo-sp-frame-access
--stress-turbo-late-spilling (optimize placement of all spill instructions, not just loop-topphis)
    type: bool  default: --no-stress-turbo-late-spilling
--turbo-filter (optimization filter for TurboFan compiler)
    type: string  default: --turbo-filter="*"
--trace-turbo (trace generated TurboFan IR)
    type: bool  default: --no-trace-turbo
--trace-turbo-path (directory to dump generated TurboFan IR to)
    type: string  default: --trace-turbo-path=""
--trace-turbo-filter (filter for tracing turbofan compilation)
    type: string  default: --trace-turbo-filter="*"
--trace-turbo-graph (trace generated TurboFan graphs)
    type: bool  default: --no-trace-turbo-graph
--trace-turbo-scheduled (trace TurboFan IR with schedule)
    type: bool  default: --no-trace-turbo-scheduled
--trace-turbo-file-prefix (trace turbo graph to a file with given prefix)
    type: string  default: --trace-turbo-file-prefix="turbo"
--trace-turbo-cfg-file (trace turbo cfg graph (for C1 visualizer) to a given file name)
    type: string  default: --trace-turbo-cfg-file=""
--trace-turbo-types (trace TurboFan's types)
    type: bool  default: --trace-turbo-types
--trace-turbo-scheduler (trace TurboFan's scheduler)
    type: bool  default: --no-trace-turbo-scheduler
--trace-turbo-reduction (trace TurboFan's various reducers)
    type: bool  default: --no-trace-turbo-reduction
--trace-turbo-trimming (trace TurboFan's graph trimmer)
```

```

    type: bool default: --no-trace-turbo-trimming
--trace-turbo-jt (trace TurboFan's jump threading)
    type: bool default: --no-trace-turbo-jt
--trace-turbo-ceq (trace TurboFan's control equivalence)
    type: bool default: --no-trace-turbo-ceq
--trace-turbo-loop (trace TurboFan's loop optimizations)
    type: bool default: --no-trace-turbo-loop
--trace-turbo-alloc (trace TurboFan's register allocator)
    type: bool default: --no-trace-turbo-alloc
--trace-all-uses (trace all use positions)
    type: bool default: --no-trace-all-uses
--trace-representation (trace representation types)
    type: bool default: --no-trace-representation
--trace-turbo-stack-accesses (trace stack load/store counters for optimized code in run-time (x64 only))
    type: bool default: --no-trace-turbo-stack-accesses
--turbo-verify (verify TurboFan graphs at each phase)
    type: bool default: --no-turbo-verify
--turbo-verify-machine-graph (verify TurboFan machine graph before instruction selection)
    type: string default: --turbo-verify-machine-graph=""
--verify-csa (verify TurboFan machine graph of code stubs)
    type: bool default: --no-verify-csa
--trace-verify-csa (trace code stubs verification)
    type: bool default: --no-trace-verify-csa
--csa-trap-on-node (trigger break point when a node with given id is created in given stub. The format is:
StubName,NodeId)
    type: string default: --csa-trap-on-node=""
--fixed-array-bounds-checks (enable FixedArray bounds checks)
    type: bool default: --fixed-array-bounds-checks
--turbo-stats (print TurboFan statistics)
    type: bool default: --no-turbo-stats
--turbo-stats-nvp (print TurboFan statistics in machine-readable format)
    type: bool default: --no-turbo-stats-nvp
--turbo-stats-wasm (print TurboFan statistics of wasm compilations)
    type: bool default: --no-turbo-stats-wasm
--turbo-splitting (split nodes during scheduling in TurboFan)
    type: bool default: --turbo-splitting
--turbo-inlining (enable inlining in TurboFan)
    type: bool default: --turbo-inlining
--max-inlined-bytecode-size (maximum size of bytecode for a single inlining)
    type: int default: --max-inlined-bytecode-size=460
--max-inlined-bytecode-size-cumulative (maximum cumulative size of bytecode considered for inlining)
    type: int default: --max-inlined-bytecode-size-cumulative=920
--max-inlined-bytecode-size-absolute (maximum absolute size of bytecode considered for inlining)
    type: int default: --max-inlined-bytecode-size-absolute=4600
--reserve-inline-budget-scale-factor (scale factor of bytecode size used to calculate the inlining budget)
    type: float default: --reserve-inline-budget-scale-factor=1.2
--max-inlined-bytecode-size-small (maximum size of bytecode considered for small function inlining)
    type: int default: --max-inlined-bytecode-size-small=27
--max-optimized-bytecode-size (maximum bytecode size to be considered for optimization; too high values may
cause the compiler to hit (release) assertions)
    type: int default: --max-optimized-bytecode-size=61440
--min-inlining-frequency (minimum frequency for inlining)
    type: float default: --min-inlining-frequency=0.15
--polymorphic-inlining (polymorphic inlining)
    type: bool default: --polymorphic-inlining
--stress-inline (set high thresholds for inlining to inline as much as possible)
    type: bool default: --no-stress-inline
--trace-turbo-inlining (trace TurboFan inlining)
    type: bool default: --no-trace-turbo-inlining
--turbo-inline-array-builtins (inline array builtins in TurboFan code)
    type: bool default: --turbo-inline-array-builtins
--use-osr (use on-stack replacement)
    type: bool default: --use-osr
--concurrent-osr (enable concurrent OSR)
    type: bool default: --concurrent-osr
--turbo-string-builder (use TurboFan fast string builder)
    type: bool default: --no-turbo-string-builder
--trace-osr (trace on-stack replacement)
    type: bool default: --no-trace-osr

```

```
--log-or-trace-osr (internal helper flag, please use --trace-osr instead.)
    type: bool  default: --no-log-or-trace-osr
--analyze-environment-liveness (analyze liveness of environment slots and zap dead values)
    type: bool  default: --analyze-environment-liveness
--trace-environment-liveness (trace liveness of local variable slots)
    type: bool  default: --no-trace-environment-liveness
--turbo-load-elimination (enable load elimination in TurboFan)
    type: bool  default: --turbo-load-elimination
--trace-turbo-load-elimination (trace TurboFan load elimination)
    type: bool  default: --no-trace-turbo-load-elimination
--turbo-profiling (enable basic block profiling in TurboFan)
    type: bool  default: --no-turbo-profiling
--turbo-profiling-verbose (enable basic block profiling in TurboFan, and include each function's schedule and
disassembly in the output)
    type: bool  default: --no-turbo-profiling-verbose
--turbo-profiling-output (emit data about basic block usage in builtins to this file (requires that V8 was
built with v8_enable_builtins_profiling=true))
    type: string default: --turbo-profiling-output=""
--abort-on-bad-builtin-profile-data (flag for mksnapshot, abort if builtins profile can't be applied)
    type: bool  default: --no-abort-on-bad-builtin-profile-data
--warn-about-builtin-profile-data (flag for mksnapshot, emit warnings when applying builtin profile data)
    type: bool  default: --no-warn-about-builtin-profile-data
--turbo-verify-allocation (verify register allocation in TurboFan)
    type: bool  default: --no-turbo-verify-allocation
--turbo-move-optimization (optimize gap moves in TurboFan)
    type: bool  default: --turbo-move-optimization
--turbo-jt (enable jump threading in TurboFan)
    type: bool  default: --turbo-jt
--turbo-loop-peeling (TurboFan loop peeling)
    type: bool  default: --turbo-loop-peeling
--turbo-loop-variable (TurboFan loop variable optimization)
    type: bool  default: --turbo-loop-variable
--turbo-loop-rotation (TurboFan loop rotation)
    type: bool  default: --turbo-loop-rotation
--turbo-cf-optimization (optimize control flow in TurboFan)
    type: bool  default: --turbo-cf-optimization
--turbo-escape (enable escape analysis)
    type: bool  default: --turbo-escape
--turbo-allocation-folding (TurboFan allocation folding)
    type: bool  default: --turbo-allocation-folding
--turbo-instruction-scheduling (enable instruction scheduling in TurboFan)
    type: bool  default: --no-turbo-instruction-scheduling
--turbo-stress-instruction-scheduling (randomly schedule instructions to stress dependency tracking)
    type: bool  default: --no-turbo-stress-instruction-scheduling
--turbo-store-elimination (enable store-store elimination in TurboFan)
    type: bool  default: --turbo-store-elimination
--trace-store-elimination (trace store elimination)
    type: bool  default: --no-trace-store-elimination
--turbo-rewrite-far-jumps (rewrite far to near jumps (ia32,x64))
    type: bool  default: --turbo-rewrite-far-jumps
--turbo-rab-gsab (optimize ResizableArrayBuffer / GrowableSharedArrayBuffer in TurboFan)
    type: bool  default: --turbo-rab-gsab
--stress-gc-during-compilation (simulate GC/compiler thread race related to https://crbug.com/v8/8520)
    type: bool  default: --no-stress-gc-during-compilation
--turbo-fast-api-calls (enable fast API calls from TurboFan)
    type: bool  default: --turbo-fast-api-calls
--turbo-compress-translation-arrays (compress translation arrays (experimental))
    type: bool  default: --no-turbo-compress-translation-arrays
--turbo-inline-js-wasm-calls (inline JS->Wasm calls)
    type: bool  default: --turbo-inline-js-wasm-calls
--turbo-use-mid-tier-regalloc-for-huge-functions (fall back to the mid-tier register allocator for huge
functions)
    type: bool  default: --turbo-use-mid-tier-regalloc-for-huge-functions
--turbo-force-mid-tier-regalloc (always use the mid-tier register allocator (for testing))
    type: bool  default: --no-turbo-force-mid-tier-regalloc
--turbo-optimize-apply (optimize Function.prototype.apply)
    type: bool  default: --turbo-optimize-apply
--turbo-optimize-math-minmax (optimize call math.min/max with double array)
    type: bool  default: --turbo-optimize-math-minmax
```

```
--turbo-collect-feedback-in-generic-lowering (enable experimental feedback collection in generic lowering.)
    type: bool  default: --no-turbo-collect-feedback-in-generic-lowering
--isolate-script-cache-ageing (enable ageing of the isolate script cache.)
    type: bool  default: --isolate-script-cache-ageing
--turboshift (enable TurboFan's Turboshift phases for JS (experimental))
    type: bool  default: --no-turboshift
--turboshift-trace-reduction (trace individual Turboshift reduction steps)
    type: bool  default: --no-turboshift-trace-reduction
--turboshift-wasm (enable TurboFan's Turboshift phases for wasm (experimental))
    type: bool  default: --no-turboshift-wasm
--optimize-for-size (Enables optimizations which favor memory size over execution speed)
    type: bool  default: --no-optimize-for-size
--wasm-generic-wrapper (allow use of the generic js-to-wasm wrapper instead of per-signature wrappers)
    type: bool  default: --wasm-generic-wrapper
--enable-wasm-arm64-generic-wrapper (allow use of the generic js-to-wasm wrapper instead of per-signature
wrappers on arm64)
    type: bool  default: --enable-wasm-arm64-generic-wrapper
--expose-wasm (expose wasm interface to JavaScript)
    type: bool  default: --expose-wasm
--wasm-num-compilation-tasks (maximum number of parallel compilation tasks for wasm)
    type: int   default: --wasm-num-compilation-tasks=128
--trace-wasm-native-heap (trace wasm native heap events)
    type: bool  default: --no-trace-wasm-native-heap
--wasm-memory-protection-keys (protect wasm code memory with PKU if available)
    type: bool  default: --wasm-memory-protection-keys
--trace-wasm-serialization (trace serialization/deserialization)
    type: bool  default: --no-trace-wasm-serialization
--wasm-async-compilation (enable actual asynchronous compilation for WebAssembly.compile)
    type: bool  default: --wasm-async-compilation
--wasm-test-streaming (use streaming compilation instead of async compilation for tests)
    type: bool  default: --no-wasm-test-streaming
--wasm-native-module-cache-enabled (enable the native module cache)
    type: bool  default: --wasm-native-module-cache-enabled
--wasm-max-mem-pages (maximum number of 64KiB memory pages per wasm memory)
    type: uint  default: --wasm-max-mem-pages=4294967295
--wasm-max-table-size (maximum table size of a wasm instance)
    type: uint  default: --wasm-max-table-size=10000000
--wasm-max-committed-code-mb (maximum committed code space for wasm (in MB))
    type: uint  default: --wasm-max-committed-code-mb=4095
--wasm-max-code-space-size-mb (maximum size of a single wasm code space)
    type: uint  default: --wasm-max-code-space-size-mb=1024
--wasm-tier-up (enable tier up to the optimizing compiler (requires --liftoff to have an effect))
    type: bool  default: --wasm-tier-up
--wasm-dynamic-tiering (enable dynamic tier up to the optimizing compiler)
    type: bool  default: --wasm-dynamic-tiering
--wasm-tiering-budget (budget for dynamic tiering (rough approximation of bytes executed)
    type: int   default: --wasm-tiering-budget=1800000
--max-wasm-functions (maximum number of wasm functions supported in a module)
    type: int   default: --max-wasm-functions=1000000
--wasm-caching-threshold (the amount of wasm top tier code that triggers the next caching event)
    type: int   default: --wasm-caching-threshold=1000000
--trace-wasm-compilation-times (print how long it took to compile each wasm function)
    type: bool  default: --no-trace-wasm-compilation-times
--wasm-tier-up-filter (only tier-up function with this index)
    type: int   default: --wasm-tier-up-filter=-1
--trace-wasm-decoder (trace decoding of wasm code)
    type: bool  default: --no-trace-wasm-decoder
--trace-wasm-compiler (trace compiling of wasm code)
    type: bool  default: --no-trace-wasm-compiler
--trace-wasm-interpreter (trace interpretation of wasm code)
    type: bool  default: --no-trace-wasm-interpreter
--trace-wasm-streaming (trace streaming compilation of wasm code)
    type: bool  default: --no-trace-wasm-streaming
--trace-wasm-stack-switching (trace wasm stack switching)
    type: bool  default: --no-trace-wasm-stack-switching
--wasm-stack-switching-stack-size (default size of stacks for wasm stack-switching (in kB))
    type: int   default: --wasm-stack-switching-stack-size=984
--liftoff (enable Liftoff, the baseline compiler for WebAssembly)
    type: bool  default: --liftoff
```

```
--liftoff-only (disallow TurboFan compilation for WebAssembly (for testing))
    type: bool  default: --no-liftoff-only
--enable-testing-opcode-in-wasm (enables a testing opcode in wasm that is only implemented in TurboFan)
    type: bool  default: --no-enable-testing-opcode-in-wasm
--trace-liftoff (trace Liftoff, the baseline compiler for WebAssembly)
    type: bool  default: --no-trace-liftoff
--trace-wasm-memory (print all memory updates performed in wasm code)
    type: bool  default: --no-trace-wasm-memory
--wasm-tier-mask-for-testing (bitmask of functions to compile with TurboFan instead of Liftoff)
    type: int   default: --wasm-tier-mask-for-testing=0
--wasm-debug-mask-for-testing (bitmask of functions to compile for debugging, only applies if the tier is Liftoff)
    type: int   default: --wasm-debug-mask-for-testing=0
--experimental-wasm-pgo-to-file (experimental: dump Wasm PGO information to a local file (for testing))
    type: bool  default: --no-experimental-wasm-pgo-to-file
--experimental-wasm-pgo-from-file (experimental: read and use Wasm PGO data from a local file (for testing))
    type: bool  default: --no-experimental-wasm-pgo-from-file
--validate-asm (validate asm.js modules before compiling)
    type: bool  default: --validate-asm
--suppress-asm-messages (don't emit asm.js related messages (for golden file testing))
    type: bool  default: --no-suppress-asm-messages
--trace-asm-time (print asm.js timing info to the console)
    type: bool  default: --no-trace-asm-time
--trace-asm-scanner (print tokens encountered by asm.js scanner)
    type: bool  default: --no-trace-asm-scanner
--trace-asm-parser (verbose logging of asm.js parse failures)
    type: bool  default: --no-trace-asm-parser
--stress-validate-asm (try to validate everything as asm.js)
    type: bool  default: --no-stress-validate-asm
--dump-wasm-module (dump wasm module bytes)
    type: bool  default: --no-dump-wasm-module
--dump-wasm-module-path (directory to dump wasm modules to)
    type: string default: --dump-wasm-module-path=""
--experimental-wasm-compilation-hints (enable compilation hints section for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-compilation-hints
--experimental-wasm-instruction-tracing (enable instruction tracing section for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-instruction-tracing
--experimental-wasm-gc (enable garbage collection for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-gc
--experimental-wasm-js-inlining (enable inline small wasm functions into JS for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-js-inlining
--experimental-wasm-assume-ref-cast-succeeds (enable assume ref.cast always succeeds and skip the related type
check (unsafe) for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-assume-ref-cast-succeeds
--experimental-wasm-ref-cast-nop (enable enable unsafe ref.cast_nop instruction for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-ref-cast-nop
--experimental-wasm-skip-null-checks (enable skip null checks for call.ref and array and struct operations
(unsafe) for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-skip-null-checks
--experimental-wasm-skip-bounds-checks (enable skip array bounds checks (unsafe) for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-skip-bounds-checks
--experimental-wasm-typed-funcref (enable typed function references for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-typed-funcref
--experimental-wasm-branch-hinting (enable branch hinting for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-branch-hinting
--experimental-wasm-stack-switching (enable stack switching for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-stack-switching
--experimental-wasm-stringref (enable reference-typed strings for Wasm (experimental))
    type: bool  default: --no-experimental-wasm-stringref
--experimental-wasm-type-reflection (enable wasm type reflection in JS for Wasm)
    type: bool  default: --no-experimental-wasm-type-reflection
--experimental-wasm-memory64 (enable memory64 for Wasm)
    type: bool  default: --no-experimental-wasm-memory64
--experimental-wasm-relaxed-simd (enable relaxed simd for Wasm)
    type: bool  default: --no-experimental-wasm-relaxed-simd
--experimental-wasm-extended-const (enable extended constant expressions for Wasm)
    type: bool  default: --no-experimental-wasm-extended-const
--experimental-wasm-inlining (enable wasm-into-wasm inlining for Wasm)
    type: bool  default: --no-experimental-wasm-inlining
```

```
--experimental-wasm-return-call (enable return call opcodes for Wasm)
    type: bool default: --experimental-wasm-return-call
--wasm-staging (enable staged wasm features)
    type: bool default: --no-wasm-staging
--wasm-opt (enable wasm optimization)
    type: bool default: --wasm-opt
--wasm-bounds-checks (enable bounds checks (disable for performance testing only))
    type: bool default: --wasm-bounds-checks
--wasm-stack-checks (enable stack checks (disable for performance testing only))
    type: bool default: --wasm-stack-checks
--wasm-enforce-bounds-checks (enforce explicit bounds check even if the trap handler is available)
    type: bool default: --no-wasm-enforce-bounds-checks
--wasm-math-intrinsics (intrinsify some Math imports into wasm)
    type: bool default: --wasm-math-intrinsics
--wasm-inlining (enable inlining of wasm functions into wasm functions (experimental))
    type: bool default: --no-wasm-inlining
--wasm-inlining-budget (maximum graph size (in TF nodes) that allows inlining more)
    type: size_t default: --wasm-inlining-budget=5000
--wasm-inlining-max-size (maximum function size (in wire bytes) that may be inlined)
    type: size_t default: --wasm-inlining-max-size=500
--wasm-speculative-inlining (enable speculative inlining of call_ref targets (experimental))
    type: bool default: --no-wasm-speculative-inlining
--trace-wasm-inlining (trace wasm inlining)
    type: bool default: --no-trace-wasm-inlining
--trace-wasm-speculative-inlining (trace wasm speculative inlining)
    type: bool default: --no-trace-wasm-speculative-inlining
--trace-wasm-typer (trace wasm typer)
    type: bool default: --no-trace-wasm-typer
--wasm-final-types (enable final types as default for wasm-gc)
    type: bool default: --no-wasm-final-types
--wasm-loop-unrolling (enable loop unrolling for wasm functions)
    type: bool default: --wasm-loop-unrolling
--wasm-loop-peeling (enable loop peeling for wasm functions)
    type: bool default: --wasm-loop-peeling
--wasm-loop-peeling-max-size (maximum size for peeling)
    type: size_t default: --wasm-loop-peeling-max-size=1000
--trace-wasm-loop-peeling (trace wasm loop peeling)
    type: bool default: --no-trace-wasm-loop-peeling
--wasm-fuzzer-gen-test (generate a test case when running a wasm fuzzer)
    type: bool default: --no-wasm-fuzzer-gen-test
--print-wasm-code (print WebAssembly code)
    type: bool default: --no-print-wasm-code
--print-wasm-code-function-index (print WebAssembly code for function at index)
    type: int default: --print-wasm-code-function-index=-1
--print-wasm-stub-code (print WebAssembly stub code)
    type: bool default: --no-print-wasm-stub-code
--asm-wasm-lazy-compilation (enable lazy compilation for asm-wasm modules)
    type: bool default: --asm-wasm-lazy-compilation
--wasm-lazy-compilation (enable lazy compilation for all wasm modules)
    type: bool default: --wasm-lazy-compilation
--trace-wasm-lazy-compilation (trace lazy compilation of wasm functions)
    type: bool default: --no-trace-wasm-lazy-compilation
--wasm-lazy-validation (enable lazy validation for lazily compiled wasm functions)
    type: bool default: --no-wasm-lazy-validation
--wasm-simd-ssse3-codegen (allow wasm SIMD SSSE3 codegen)
    type: bool default: --no-wasm-simd-ssse3-codegen
--wasm-code-gc (enable garbage collection of wasm code)
    type: bool default: --wasm-code-gc
--trace-wasm-code-gc (trace garbage collection of wasm code)
    type: bool default: --no-trace-wasm-code-gc
--stress-wasm-code-gc (stress test garbage collection of wasm code)
    type: bool default: --no-stress-wasm-code-gc
--wasm-max-initial-code-space-reservation (maximum size of the initial wasm code space reservation (in MB))
    type: int default: --wasm-max-initial-code-space-reservation=0
--wasm-max-module-size (maximum allowed size of wasm modules)
    type: size_t default: --wasm-max-module-size=1073741824
--wasm-disassembly-max-mb (maximum size of produced disassembly (in MB, approximate))
    type: size_t default: --wasm-disassembly-max-mb=1000
--trace-wasm (trace wasm function calls)
```

```
    type: bool  default: --no-trace-wasm
--trace-wasm-instances (trace creation and collection of wasm instances)
    type: bool  default: --no-trace-wasm-instances
--stress-sampling-allocation-profiler (Enables sampling allocation profiler with X as a sample interval)
    type: int   default: --stress-sampling-allocation-profiler=0
--lazy-new-space-shrinking (Enables the lazy new space shrinking strategy)
    type: bool  default: --no-lazy-new-space-shrinking
--min-semi-space-size (min size of a semi-space (in MBytes), the new space consists of two semi-spaces)
    type: size_t default: --min-semi-space-size=0
--max-semi-space-size (max size of a semi-space (in MBytes), the new space consists of two semi-spaces)
    type: size_t default: --max-semi-space-size=0
--semi-space-growth-factor (factor by which to grow the new space)
    type: int   default: --semi-space-growth-factor=2
--max-old-space-size (max size of the old space (in Mbytes))
    type: size_t default: --max-old-space-size=0
--max-heap-size (max size of the heap (in Mbytes) both max_semi_space_size and max_old_space_size take
precedence. All three flags cannot be specified at the same time.)
    type: size_t default: --max-heap-size=0
--initial-heap-size (initial size of the heap (in Mbytes))
    type: size_t default: --initial-heap-size=0
--huge-max-old-generation-size (Increase max size of the old space to 4 GB for x64 systems with the physical
memory bigger than 16 GB)
    type: bool  default: --huge-max-old-generation-size
--initial-old-space-size (initial old space size (in Mbytes))
    type: size_t default: --initial-old-space-size=0
--separate-gc-phases (young and full garbage collection phases are not overlapping)
    type: bool  default: --no-separate-gc-phases
--gc-global (always perform global GCs)
    type: bool  default: --no-gc-global
--random-gc-interval (Collect garbage after random(0, X) allocations. It overrides gc_interval.)
    type: int   default: --random-gc-interval=0
--gc-interval (garbage collect after <n> allocations)
    type: int   default: --gc-interval=-1
--retain-maps-for-n-gc (keeps maps alive for <n> old space garbage collections)
    type: int   default: --retain-maps-for-n-gc=2
--trace-gc (print one trace line following each garbage collection)
    type: bool  default: --no-trace-gc
--trace-gc-nvp (print one detailed trace line in name=value format after each garbage collection)
    type: bool  default: --no-trace-gc-nvp
--trace-gc-ignore-scraper (do not print trace line after scraper collection)
    type: bool  default: --no-trace-gc-ignore-scraper
--trace-idle-notification (print one trace line following each idle notification)
    type: bool  default: --no-trace-idle-notification
--trace-idle-notification-verbose (prints the heap state used by the idle notification)
    type: bool  default: --no-trace-idle-notification-verbose
--trace-gc-verbose (print more details following each garbage collection)
    type: bool  default: --no-trace-gc-verbose
--trace-gc-freelists (prints details of each freelist before and after each major garbage collection)
    type: bool  default: --no-trace-gc-freelists
--trace-gc-freelists-verbose (prints details of freelists of each page before and after each major garbage
collection)
    type: bool  default: --no-trace-gc-freelists-verbose
--trace-gc-heap-layout (print layout of pages in heap before and after gc)
    type: bool  default: --no-trace-gc-heap-layout
--trace-gc-heap-layout-ignore-minor-gc (do not print trace line before and after minor-gc)
    type: bool  default: --trace-gc-heap-layout-ignore-minor-gc
--trace-evacuation-candidates (Show statistics about the pages evacuation by the compaction)
    type: bool  default: --no-trace-evacuation-candidates
--trace-allocations-origins (Show statistics about the origins of allocations. Combine with --no-inline-new to
track allocations from generated code)
    type: bool  default: --no-trace-allocations-origins
--trace-pending-allocations (trace calls to Heap::IsAllocationPending that return true)
    type: bool  default: --no-trace-pending-allocations
--trace-allocation-stack-interval (print stack trace after <n> free-list allocations)
    type: int   default: --trace-allocation-stack-interval=-1
--trace-duplicate-threshold-kb (print duplicate objects in the heap if their size is more than given threshold)
    type: int   default: --trace-duplicate-threshold-kb=0
--trace-fragmentation (report fragmentation for old space)
    type: bool  default: --no-trace-fragmentation
```

```
--trace-fragmentation-verbose (report fragmentation for old space (detailed))
    type: bool default: --no-trace-fragmentation-verbose
--minor-mc-trace-fragmentation (trace fragmentation after marking)
    type: bool default: --no-minor-mc-trace-fragmentation
--trace-evacuation (report evacuation statistics)
    type: bool default: --no-trace-evacuation
--trace-mutator-utilization (print mutator utilization, allocation speed, gc speed)
    type: bool default: --no-trace-mutator-utilization
--incremental-marking (use incremental marking)
    type: bool default: --incremental-marking
--incremental-marking-wrappers (use incremental marking for marking wrappers)
    type: bool default: --incremental-marking-wrappers
--incremental-marking-task (use tasks for incremental marking)
    type: bool default: --incremental-marking-task
--incremental-marking-soft-trigger (threshold for starting incremental marking via a task in percent of
available space: limit - size)
    type: int default: --incremental-marking-soft-trigger=0
--fast-forward-schedule (Fast forwards marking schedule)
    type: bool default: --no-fast-forward-schedule
--incremental-marking-hard-trigger (threshold for starting incremental marking immediately in percent of
available space: limit - size)
    type: int default: --incremental-marking-hard-trigger=0
--trace-unmapper (Trace the unmapping)
    type: bool default: --no-trace-unmapper
--parallel-scavenge (parallel scavenge)
    type: bool default: --parallel-scavenge
--minor-gc-task (schedule scavenge tasks)
    type: bool default: --minor-gc-task
--minor-gc-task-trigger (minor GC task trigger in percent of the current heap limit)
    type: int default: --minor-gc-task-trigger=80
--scavenge-separate-stack-scanning (use a separate phase for stack scanning in scavenge)
    type: bool default: --no-scavenge-separate-stack-scanning
--trace-parallel-scavenge (trace parallel scavenge)
    type: bool default: --no-trace-parallel-scavenge
--cppgc-young-generation (run young generation garbage collections in Oilpan (experimental))
    type: bool default: --no-cppgc-young-generation
--write-protect-code-memory (write protect code memory)
    type: bool default: --no-write-protect-code-memory
--concurrent-marking (use concurrent marking)
    type: bool default: --concurrent-marking
--concurrent-marking-max-worker-num (max worker number of concurrent marking, 0 for NumberOfWorkerThreads)
    type: int default: --concurrent-marking-max-worker-num=7
--concurrent-array-buffer-sweeping (concurrently sweep array buffers)
    type: bool default: --concurrent-array-buffer-sweeping
--stress-concurrent-allocation (start background threads that allocate memory)
    type: bool default: --no-stress-concurrent-allocation
--parallel-marking (use parallel marking in atomic pause)
    type: bool default: --parallel-marking
--ephemeron-fixpoint-iterations (number of fixpoint iterations it takes to switch to linear ephemeron
algorithm)
    type: int default: --ephemeron-fixpoint-iterations=10
--trace-concurrent-marking (trace concurrent marking)
    type: bool default: --no-trace-concurrent-marking
--concurrent-sweeping (use concurrent sweeping)
    type: bool default: --concurrent-sweeping
--parallel-compaction (use parallel compaction)
    type: bool default: --parallel-compaction
--parallel-pointer-update (use parallel pointer update during compaction)
    type: bool default: --parallel-pointer-update
--detect-ineffective-gcs-near-heap-limit (trigger out-of-memory failure to avoid GC storm near heap limit)
    type: bool default: --detect-ineffective-gcs-near-heap-limit
--trace-incremental-marking (trace progress of the incremental marking)
    type: bool default: --no-trace-incremental-marking
--trace-stress-marking (trace stress marking progress)
    type: bool default: --no-trace-stress-marking
--trace-stress-scavenge (trace stress scavenge progress)
    type: bool default: --no-trace-stress-scavenge
--track-gc-object-stats (track object counts and memory usage)
    type: bool default: --no-track-gc-object-stats
```

```
--trace-gc-object-stats (trace object counts and memory usage)
    type: bool  default: --no-trace-gc-object-stats
--trace-zone-stats (trace zone memory usage)
    type: bool  default: --no-trace-zone-stats
--zone-stats-tolerance (report a tick only when allocated zone memory changes by this amount)
    type: size_t default: --zone-stats-tolerance=1048576
--trace-zone-type-stats (trace per-type zone memory usage)
    type: bool  default: --no-trace-zone-type-stats
--track-retaining-path (enable support for tracking retaining path)
    type: bool  default: --no-track-retaining-path
--trace-backing-store (trace backing store events)
    type: bool  default: --no-trace-backing-store
--gc-stats (Used by tracing internally to enable gc statistics)
    type: int   default: --gc-stats=0
--track-detached-contexts (track native contexts that are expected to be garbage collected)
    type: bool  default: --track-detached-contexts
--trace-detached-contexts (trace native contexts that are expected to be garbage collected)
    type: bool  default: --no-trace-detached-contexts
--verify-heap (verify heap pointers before and after GC)
    type: bool  default: --no-verify-heap
--move-object-start (enable moving of object starts)
    type: bool  default: --move-object-start
--memory-reducer (use memory reducer)
    type: bool  default: --memory-reducer
--memory-reducer-for-small-heaps (use memory reducer for small heaps)
    type: bool  default: --memory-reducer-for-small-heaps
--memory-reducer-single-gc (only schedule a single GC from memory reducer)
    type: bool  default: --no-memory-reducer-single-gc
--heap-growing-percent (specifies heap growing factor as (1 + heap_growing_percent/100))
    type: int   default: --heap-growing-percent=0
--v8-os-page-size (override OS page size (in KBytes))
    type: int   default: --v8-os-page-size=0
--allocation-buffer-parking (allocation buffer parking)
    type: bool  default: --allocation-buffer-parking
--compact (Perform compaction on full GCs based on V8's default heuristics)
    type: bool  default: --compact
--compact-code-space (Perform code space compaction on full collections.)
    type: bool  default: --compact-code-space
--compact-on-every-full-gc (Perform compaction on every full GC)
    type: bool  default: --no-compact-on-every-full-gc
--compact-with-stack (Perform compaction when finalizing a full GC with stack)
    type: bool  default: --compact-with-stack
--compact-code-space-with-stack (Perform code space compaction when finalizing a full GC with stack)
    type: bool  default: --compact-code-space-with-stack
--shortcut-strings-with-stack (Shortcut Strings during GC with stack)
    type: bool  default: --shortcut-strings-with-stack
--stress-compaction (Stress GC compaction to flush out bugs (implies --force_marking_deque_overflows))
    type: bool  default: --no-stress-compaction
--stress-compaction-random (Stress GC compaction by selecting random percent of pages as evacuation candidates.
Overrides stress_compaction.)
    type: bool  default: --no-stress-compaction-random
--flush-baseline-code (flush of baseline code when it has not been executed recently)
    type: bool  default: --no-flush-baseline-code
--flush-bytecode (flush of bytecode when it has not been executed recently)
    type: bool  default: --flush-bytecode
--bytecode-old-age (number of gcs before we flush code)
    type: int   default: --bytecode-old-age=5
--stress-flush-code (stress code flushing)
    type: bool  default: --no-stress-flush-code
--trace-flush-bytecode (trace bytecode flushing)
    type: bool  default: --no-trace-flush-bytecode
--use-marking-progress-bar (Use a progress bar to scan large objects in increments when incremental marking is
active.)
    type: bool  default: --use-marking-progress-bar
--stress-per-context-marking-worklist (Use per-context worklist for marking)
    type: bool  default: --no-stress-per-context-marking-worklist
--force-marking-deque-overflows (force overflows of marking deque by reducing it's size to 64 words)
    type: bool  default: --no-force-marking-deque-overflows
--stress-incremental-marking (force incremental marking for small heaps and run it more often)
```

```
    type: bool  default: --no-stress-incremental-marking
--fuzzer-gc-analysis (prints number of allocations and enables analysis mode for gc fuzz testing, e.g.
--stress-marking, --stress-scavenge)
    type: bool  default: --no-fuzzer-gc-analysis
--stress-marking (force marking at random points between 0 and X (inclusive) percent of the regular marking
start limit)
    type: int   default: --stress-marking=0
--stress-scavenge (force scavenge at random points between 0 and X (inclusive) percent of the new space
capacity)
    type: int   default: --stress-scavenge=0
--reclaim-unmodified-wrappers (reclaim otherwise unreachable unmodified wrapper objects when possible)
    type: bool  default: --reclaim-unmodified-wrappers
--gc-experiment-less-compaction (less compaction in non-memory reducing mode)
    type: bool  default: --no-gc-experiment-less-compaction
--gc-memory-reducer-start-delay-ms (Delay before memory reducer start)
    type: int   default: --gc-memory-reducer-start-delay-ms=8000
--disable-abortjs (disables AbortJS runtime function)
    type: bool  default: --no-disable-abortjs
--randomize-all-allocations (randomize virtual memory reservations by ignoring any hints passed when allocating
pages)
    type: bool  default: --no-randomize-all-allocations
--manual-evacuation-candidates-selection (Test mode only flag. It allows an unit test to select evacuation
candidates pages (requires --stress_compaction).)
    type: bool  default: --no-manual-evacuation-candidates-selection
--clear-free-memory (initialize free memory with 0)
    type: bool  default: --no-clear-free-memory
--crash-on-aborted-evacuation (crash when evacuation of page fails)
    type: bool  default: --no-crash-on-aborted-evacuation
--cppheap-incremental-marking (use incremental marking for CppHeap)
    type: bool  default: --cppheap-incremental-marking
--cppheap-concurrent-marking (use concurrent marking for CppHeap)
    type: bool  default: --cppheap-concurrent-marking
--debug-code ()
    type: bool  default: --no-debug-code
--code-comments ()
    type: bool  default: --no-code-comments
--enable-sse3 (enable use of SSE3 instructions if available)
    type: bool  default: --enable-sse3
--enable-ssse3 (enable use of SSSE3 instructions if available)
    type: bool  default: --enable-ssse3
--enable-sse4-1 (enable use of SSE4.1 instructions if available)
    type: bool  default: --enable-sse4-1
--enable-sse4-2 (enable use of SSE4.2 instructions if available)
    type: bool  default: --enable-sse4-2
--enable-sahf (enable use of SAHF instruction if available (X64 only))
    type: bool  default: --enable-sahf
--enable-avx (enable use of AVX instructions if available)
    type: bool  default: --enable-avx
--enable-avx2 (enable use of AVX2 instructions if available)
    type: bool  default: --enable-avx2
--enable-fma3 (enable use of FMA3 instructions if available)
    type: bool  default: --enable-fma3
--enable-bmi1 (enable use of BMI1 instructions if available)
    type: bool  default: --enable-bmi1
--enable-bmi2 (enable use of BMI2 instructions if available)
    type: bool  default: --enable-bmi2
--enable-lzcnt (enable use of LZCNT instruction if available)
    type: bool  default: --enable-lzcnt
--enable-popcnt (enable use of POPCNT instruction if available)
    type: bool  default: --enable-popcnt
--arm-arch (generate instructions for the selected ARM architecture if available: armv6, armv7, armv7+sudiv or
armv8)
    type: string default: --arm-arch="armv8"
--force-long-branches (force all emitted branches to be in long mode (MIPS/PPC only))
    type: bool  default: --no-force-long-branches
--mcpu (enable optimization for specific cpu)
    type: string default: --mcpu="auto"
--partial-constant-pool (enable use of partial constant pools (X64 only))
    type: bool  default: --partial-constant-pool
```

```
--sim-arm64-optional-features (enable optional features on the simulator for testing: none or all)
    type: string default: --sim-arm64-optional-features="none"
--enable-source-at-csa-bind (Include source information in the binary at CSA bind locations.)
    type: bool default: --no-enable-source-at-csa-bind
--enable-armv7 (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-armv7=unset
--enable-vfp3 (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-vfp3=unset
--enable-32dregs (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-32dregs=unset
--enable-neon (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-neon=unset
--enable-sudiv (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-sudiv=unset
--enable-armv8 (deprecated (use --arm_arch instead))
    type: maybe_bool default: --enable-armv8=unset
--enable-regexp-unaligned-accesses (enable unaligned accesses for the regexp engine)
    type: bool default: --enable-regexp-unaligned-accesses
--script-streaming (enable parsing on background)
    type: bool default: --script-streaming
--stress-background-compile (stress test parsing on background)
    type: bool default: --no-stress-background-compile
--concurrent-cache-deserialization (enable deserializing code caches on background)
    type: bool default: --concurrent-cache-deserialization
--merge-background-deserialized-script-with-compilation-cache (After deserializing code cache data on a
background thread, merge it into an existing Script if one is found in the Isolate compilation cache)
    type: bool default: --merge-background-deserialized-script-with-compilation-cache
--disable-old-api-accessors (Disable old-style API accessors whose setters trigger through the prototype chain)
    type: bool default: --no-disable-old-api-accessors
--embedder-instance-types (enable type checks based on instance types provided by the embedder)
    type: bool default: --no-embedder-instance-types
--expose-gc (expose gc extension)
    type: bool default: --no-expose-gc
--expose-gc-as (expose gc extension under the specified name)
    type: string default: --expose-gc-as=""
--expose-externalize-string (expose externalize string extension)
    type: bool default: --no-expose-externalize-string
--expose-statistics (expose statistics extension)
    type: bool default: --no-expose-statistics
--expose-trigger-failure (expose trigger-failure extension)
    type: bool default: --no-expose-trigger-failure
--expose-ignition-statistics (expose ignition-statistics extension (requires building with
v8_enable_ignition_dispatch_counting))
    type: bool default: --no-expose-ignition-statistics
--stack-trace-limit (number of stack frames to capture)
    type: int default: --stack-trace-limit=10
--builtins-in-stack-traces (show built-in functions in stack traces)
    type: bool default: --no-builtins-in-stack-traces
--experimental-stack-trace-frames (enable experimental frames (API/Builtins) and stack trace layout)
    type: bool default: --no-experimental-stack-trace-frames
--disallow-code-generation-from-strings (disallow eval and friends)
    type: bool default: --no-disallow-code-generation-from-strings
--expose-async-hooks (expose async_hooks object)
    type: bool default: --no-expose-async-hooks
--expose-cputracemark-as (expose cputracemark extension under the specified name)
    type: string default: --expose-cputracemark-as=""
--allow-unsafe-function-constructor (allow invoking the function constructor without security checks)
    type: bool default: --no-allow-unsafe-function-constructor
--force-slow-path (always take the slow path for builtins)
    type: bool default: --no-force-slow-path
--test-small-max-function-context-stub-size (enable testing the function context size overflow path by making
the maximum size smaller)
    type: bool default: --no-test-small-max-function-context-stub-size
--inline-new (use fast inline allocation)
    type: bool default: --inline-new
--switch-table-spread-threshold (allow the jump table used for switch statements to span a range of integers
roughly equal to this number times the number of clauses in the switch)
    type: int default: --switch-table-spread-threshold=3
--switch-table-min-cases (the number of Smi integer cases present in the switch statement before using the jump
```

```
table optimization)
    type: int  default: --switch-table-min-cases=6
--trace (trace javascript function calls)
    type: bool  default: --no-trace
--lazy (use lazy compilation)
    type: bool  default: --lazy
--lazy-eval (use lazy compilation during eval)
    type: bool  default: --lazy-eval
--lazy-streaming (use lazy compilation during streaming compilation)
    type: bool  default: --lazy-streaming
--max-lazy (ignore eager compilation hints)
    type: bool  default: --no-max-lazy
--trace-opt (trace optimized compilation)
    type: bool  default: --no-trace-opt
--trace-opt-verbose (extra verbose optimized compilation tracing)
    type: bool  default: --no-trace-opt-verbose
--trace-opt-stats (trace optimized compilation statistics)
    type: bool  default: --no-trace-opt-stats
--trace-deopt (trace deoptimization)
    type: bool  default: --no-trace-deopt
--log-deopt (log deoptimization)
    type: bool  default: --no-log-deopt
--trace-deopt-verbose (extra verbose deoptimization tracing)
    type: bool  default: --no-trace-deopt-verbose
--trace-file-names (include file names in trace-opt/trace-deopt output)
    type: bool  default: --no-trace-file-names
--always-turbofan (always try to optimize functions)
    type: bool  default: --no-always-turbofan
--always-osr (always try to OSR functions)
    type: bool  default: --no-always-osr
--prepare-always-turbofan (prepare for turning on always opt)
    type: bool  default: --no-prepare-always-turbofan
--deopt-to-baseline (deoptimize to baseline code when available)
    type: bool  default: --no-deopt-to-baseline
--trace-serializer (print code serializer trace)
    type: bool  default: --no-trace-serializer
--compilation-cache (enable compilation cache)
    type: bool  default: --compilation-cache
--cache-prototype-transitions (cache prototype transitions)
    type: bool  default: --cache-prototype-transitions
--lazy-compile-dispatcher (enable compiler dispatcher)
    type: bool  default: --no-lazy-compile-dispatcher
--lazy-compile-dispatcher-max-threads (max threads for compiler dispatcher (0 for unbounded))
    type: uint  default: --lazy-compile-dispatcher-max-threads=0
--trace-compiler-dispatcher (trace compiler dispatcher activity)
    type: bool  default: --no-trace-compiler-dispatcher
--parallel-compile-tasks-for-eager-toplevel (spawn parallel compile tasks for eagerly compiled, top-level
functions)
    type: bool  default: --no-parallel-compile-tasks-for-eager-toplevel
--parallel-compile-tasks-for-lazy (spawn parallel compile tasks for all lazily compiled functions)
    type: bool  default: --no-parallel-compile-tasks-for-lazy
--cpu-profiler-sampling-interval (CPU profiler sampling interval in microseconds)
    type: int  default: --cpu-profiler-sampling-interval=1000
--trace-side-effect-free-debug-evaluate (print debug messages for side-effect-free debug-evaluate for testing)
    type: bool  default: --no-trace-side-effect-free-debug-evaluate
--hard-abort (abort by crashing)
    type: bool  default: --hard-abort
--experimental-value-unavailable (enable experimental <value unavailable> in scopes)
    type: bool  default: --no-experimental-value-unavailable
--log-colour (When logging, try to use coloured output.)
    type: bool  default: --no-log-colour
--expose-inspector-scripts (expose injected-script-source.js for debugging)
    type: bool  default: --no-expose-inspector-scripts
--stack-size (default size of stack region v8 is allowed to use (in kBytes))
    type: int  default: --stack-size=984
--max-stack-trace-source-length (maximum length of function source code printed in a stack trace.)
    type: int  default: --max-stack-trace-source-length=300
--clear-exceptions-on-js-entry (clear pending exceptions when entering JavaScript)
    type: bool  default: --no-clear-exceptions-on-js-entry
```

```
--histogram-interval (time interval in ms for aggregating memory histograms)
    type: int  default: --histogram-interval=600000
--heap-profiler-trace-objects (Dump heap object allocations/movements/size_updates)
    type: bool  default: --no-heap-profiler-trace-objects
--heap-profiler-use-embedder-graph (Use the new EmbedderGraph API to get embedder nodes)
    type: bool  default: --heap-profiler-use-embedder-graph
--heap-snapshot-string-limit (truncate strings to this length in the heap snapshot)
    type: int  default: --heap-snapshot-string-limit=1024
--heap-profiler-show-hidden-objects (use 'native' rather than 'hidden' node type in snapshot)
    type: bool  default: --no-heap-profiler-show-hidden-objects
--sampling-heap-profiler-suppress-randomness (Use constant sample intervals to eliminate test flakiness)
    type: bool  default: --no-sampling-heap-profiler-suppress-randomness
--use-idle-notification (Use idle notification to reduce memory footprint.)
    type: bool  default: --use-idle-notification
--log-ic (Log inline cache state transitions for tools/ic-processor)
    type: bool  default: --no-log-ic
--fast-map-update (enable fast map update by caching the migration target)
    type: bool  default: --no-fast-map-update
--max-valid-polymorphic-map-count (maximum number of valid maps to track in POLYMORPHIC state)
    type: int  default: --max-valid-polymorphic-map-count=4
--native-code-counters (generate extra code for manipulating stats counters)
    type: bool  default: --no-native-code-counters
--super-ic (use an IC for super property loads)
    type: bool  default: --super-ic
--mega-dom-ic (use MegaDOM IC state for API objects)
    type: bool  default: --no-mega-dom-ic
--trace-prototype-users (Trace updates to prototype user tracking)
    type: bool  default: --no-trace-prototype-users
--trace-for-in-enumerate (Trace for-in enumerate slow-paths)
    type: bool  default: --no-trace-for-in-enumerate
--log-maps (Log map creation)
    type: bool  default: --no-log-maps
--log-maps-details (Also log map details)
    type: bool  default: --log-maps-details
--allow-natives-syntax (allow natives syntax)
    type: bool  default: --no-allow-natives-syntax
--allow-natives-for-differential-fuzzing (allow only natives explicitly allowlisted for differential fuzzers)
    type: bool  default: --no-allow-natives-for-differential-fuzzing
--parse-only (only parse the sources)
    type: bool  default: --no-parse-only
--async-stack-traces (include async stack traces in Error.stack)
    type: bool  default: --async-stack-traces
--stack-trace-on-illegal (print stack trace when an illegal exception is thrown)
    type: bool  default: --no-stack-trace-on-illegal
--abort-on-uncaught-exception (abort program (dump core) when an uncaught exception is thrown)
    type: bool  default: --no-abort-on-uncaught-exception
--correctness-fuzzer-suppressions (Suppress certain unspecified behaviors to ease correctness fuzzing: Abort
program when the stack overflows or a string exceeds maximum length (as opposed to throwing RangeError). Use a
fixed suppression string for error messages.)
    type: bool  default: --no-correctness-fuzzer-suppressions
--rehash-snapshot (rehash strings from the snapshot to override the baked-in seed)
    type: bool  default: --rehash-snapshot
--hash-seed (Fixed seed to use to hash property keys (0 means random)(with snapshots this option cannot
override the baked-in seed))
    type: uint64  default: --hash-seed=0
--random-seed (Default seed for initializing random generator (0, the default, means to use system random).)
    type: int  default: --random-seed=0
--fuzzer-random-seed (Default seed for initializing fuzzer random generator (0, the default, means to use v8's
random number generator seed).)
    type: int  default: --fuzzer-random-seed=0
--trace-rail (trace RAIL mode)
    type: bool  default: --no-trace-rail
--print-all-exceptions (print exception object and stack trace on each thrown exception)
    type: bool  default: --no-print-all-exceptions
--detailed-error-stack-trace (includes arguments for each function call in the error stack frames array)
    type: bool  default: --no-detailed-error-stack-trace
--adjust-os-scheduling-parameters (adjust OS specific scheduling params for the isolate)
    type: bool  default: --adjust-os-scheduling-parameters
--experimental-flush-embedded-blob-icache (Used in an experiment to evaluate icache flushing on certain CPUs)
```

```
    type: bool  default: --experimental-flush-embedded-blob-icache
--short-builtin-calls (Put embedded builtins code into the code range for shorter builtin calls/jumps if system
has >=4GB memory)
    type: bool  default: --short-builtin-calls
--trace-code-range-allocation (Trace code range allocation process.)
    type: bool  default: --no-trace-code-range-allocation
--better-code-range-allocation (This mode tries harder to allocate code range near .text section. Works only
for configurations with external code space and shared pointer compression cage.)
    type: bool  default: --no-better-code-range-allocation
--abort-on-far-code-range (Abort if code range is allocated further away than 4GB from the.text section)
    type: bool  default: --no-abort-on-far-code-range
--runtime-call-stats (report runtime call counts and times)
    type: bool  default: --no-runtime-call-stats
--rcs (report runtime call counts and times)
    type: bool  default: --no-rcs
--rcs-cpu-time (report runtime times in cpu time (the default is wall time))
    type: bool  default: --no-rcs-cpu-time
--verify-snapshot-checksum (Verify snapshot checksums when deserializing snapshots. Enable checksum creation
and verification for code caches. Enabled by default in debug builds and once per process for Android.)
    type: bool  default: --no-verify-snapshot-checksum
--profile-deserialization (Print the time it takes to deserialize the snapshot.)
    type: bool  default: --no-profile-deserialization
--serialization-statistics (Collect statistics on serialized objects.)
    type: bool  default: --no-serialization-statistics
--regexp-optimization (generate optimized regexp code)
    type: bool  default: --regexp-optimization
--regexp-interpret-all (interpret all regexp code)
    type: bool  default: --no-regexp-interpret-all
--regexp-tier-up (enable regexp interpreter and tier up to the compiler after the number of executions set by
the tier up ticks flag)
    type: bool  default: --regexp-tier-up
--regexp-tier-up-ticks (set the number of executions for the regexp interpreter before tiering-up to the
compiler)
    type: int   default: --regexp-tier-up-ticks=1
--regexp-peephole-optimization (enable peephole optimization for regexp bytecode)
    type: bool  default: --regexp-peephole-optimization
--trace-regexp-peephole-optimization (trace regexp bytecode peephole optimization)
    type: bool  default: --no-trace-regexp-peephole-optimization
--trace-regexp-bytecodes (trace regexp bytecode execution)
    type: bool  default: --no-trace-regexp-bytecodes
--trace-regexp-assembler (trace regexp macro assembler calls.)
    type: bool  default: --no-trace-regexp-assembler
--trace-regexp-parser (trace regexp parsing)
    type: bool  default: --no-trace-regexp-parser
--trace-regexp-tier-up (trace regexp tiering up execution)
    type: bool  default: --no-trace-regexp-tier-up
--trace-regexp-graph (trace the regexp graph)
    type: bool  default: --no-trace-regexp-graph
--enable-experimental-regexp-engine (recognize regexps with 'l' flag, run them on experimental engine)
    type: bool  default: --no-enable-experimental-regexp-engine
--default-to-experimental-regexp-engine (run regexps with the experimental engine where possible)
    type: bool  default: --no-default-to-experimental-regexp-engine
--trace-experimental-regexp-engine (trace execution of experimental regexp engine)
    type: bool  default: --no-trace-experimental-regexp-engine
--enable-experimental-regexp-engine-on-excessive-backtracks (fall back to a breadth-first regexp engine on
excessive backtracking)
    type: bool  default: --no-enable-experimental-regexp-engine-on-excessive-backtracks
--regexp-backtracks-before-fallback (number of backtracks during regexp execution before fall back to
experimental engine if enable_experimental_regexp_engine_on_excessive_backtracks is set)
    type: uint  default: --regexp-backtracks-before-fallback=50000
--testing-bool-flag (testing_bool_flag)
    type: bool  default: --testing-bool-flag
--testing-maybe-bool-flag (testing_maybe_bool_flag)
    type: maybe_bool  default: --testing-maybe-bool-flag=unset
--testing-int-flag (testing_int_flag)
    type: int   default: --testing-int-flag=13
--testing-float-flag (float-flag)
    type: float  default: --testing-float-flag=2.5
--testing-string-flag (string-flag)
```

```
    type: string default: --testing-string-flag="Hello, world!"
--testing-prng-seed (Seed used for threading test randomness)
    type: int default: --testing-prng-seed=42
--testing-d8-test-runner (test runner turns on this flag to enable a check that the function was prepared for
optimization before marking it for optimization)
    type: bool default: --no-testing-d8-test-runner
--strict-termination-checks (Enable strict terminating DCHECKs to prevent accidentally keeping on executing JS
after terminating V8. (experimental))
    type: bool default: --no-strict-termination-checks
--fuzzing (Fuzzers use this flag to signal that they are ... fuzzing. This causes intrinsics to fail silently
(e.g. return undefined) on invalid usage.)
    type: bool default: --no-fuzzing
--freeze-flags-after-init (Disallow changes to flag values after initializing V8)
    type: bool default: --no-freeze-flags-after-init
--embedded-src (Path for the generated embedded data file. (mksnapshot only))
    type: string default: --embedded-src=""
--embedded-variant (Label to disambiguate symbols in embedded data file. (mksnapshot only))
    type: string default: --embedded-variant=""
--static-roots-src (Path for writing a fresh static-roots.h. (mksnapshot only, build without static roots
only))
    type: string default: --static-roots-src=""
--startup-src (Write V8 startup as C++ src. (mksnapshot only))
    type: string default: --startup-src=""
--startup-blob (Write V8 startup blob file. (mksnapshot only))
    type: string default: --startup-blob=""
--target-arch (The mksnapshot target arch. (mksnapshot only))
    type: string default: --target-arch=""
--target-os (The mksnapshot target os. (mksnapshot only))
    type: string default: --target-os=""
--target-is-simulator (Instruct mksnapshot that the target is meant to run in the simulator and it can generate
simulator-specific instructions. (mksnapshot only))
    type: bool default: --no-target-is-simulator
--turbo-profiling-input (Path of the input file containing basic block counters for builtins. (mksnapshot
only))
    type: string default: --turbo-profiling-input=""
--text-is-readable (Whether the .text section of binary can be read)
    type: bool default: --text-is-readable
--trace-minor-mc-parallel-marking (trace parallel marking for the young generation)
    type: bool default: --no-trace-minor-mc-parallel-marking
--minor-mc (perform young generation mark compact GCs (experimental))
    type: bool default: --no-minor-mc
--concurrent-minor-mc-marking (perform young generation marking concurrently (experimental))
    type: bool default: --no-concurrent-minor-mc-marking
--help (Print usage message, including flags, on console)
    type: bool default: --help
--print-flag-values (Print all flag values of V8)
    type: bool default: --no-print-flag-values
--slow-histograms (Enable slow histograms with more overhead.)
    type: bool default: --no-slow-histograms
--use-external-strings (Use external strings for source code)
    type: bool default: --no-use-external-strings
--map-counters (Map counters to a file)
    type: string default: --map-counters=""
--mock-arraybuffer-allocator (Use a mock ArrayBuffer allocator for testing.)
    type: bool default: --no-mock-arraybuffer-allocator
--mock-arraybuffer-allocator-limit (Memory limit for mock ArrayBuffer allocator used to simulate OOM for
testing.)
    type: size_t default: --mock-arraybuffer-allocator-limit=0
--gdbjit (enable GDBJIT interface)
    type: bool default: --no-gdbjit
--gdbjit-full (enable GDBJIT interface for all code objects)
    type: bool default: --no-gdbjit-full
--gdbjit-dump (dump elf objects with debug info to disk)
    type: bool default: --no-gdbjit-dump
--gdbjit-dump-filter (dump only objects containing this substring)
    type: string default: --gdbjit-dump-filter=""
--enable-slow-asserts (enable asserts that are slow to execute)
    type: bool default: --no-enable-slow-asserts
--print-ast (print source AST)
```

```
    type: bool  default: --no-print-ast
--trap-on-abort (replace aborts by breakpoints)
    type: bool  default: --no-trap-on-abort
--print-scopes (print scopes)
    type: bool  default: --no-print-scopes
--trace-contexts (trace contexts operations)
    type: bool  default: --no-trace-contexts
--gc-verbose (print stuff during garbage collection)
    type: bool  default: --no-gc-verbose
--code-stats (report code statistics after GC)
    type: bool  default: --no-code-stats
--print-handles (report handles after GC)
    type: bool  default: --no-print-handles
--check-handle-count (Check that there are not too many handles at GC)
    type: bool  default: --no-check-handle-count
--print-global-handles (report global handles after GC)
    type: bool  default: --no-print-global-handles
--trace-turbo-escape (enable tracing in escape analysis)
    type: bool  default: --no-trace-turbo-escape
--trace-module-status (Trace status transitions of ECMAScript modules)
    type: bool  default: --no-trace-module-status
--trace-normalization (prints when objects are turned into dictionaries.)
    type: bool  default: --no-trace-normalization
--trace-lazy (trace lazy compilation)
    type: bool  default: --no-trace-lazy
--trace-isolates (trace isolate state changes)
    type: bool  default: --no-trace-isolates
--regexp-possessive-quantifier (enable possessive quantifier syntax for testing)
    type: bool  default: --no-regexp-possessive-quantifier
--print-break-location (print source location on debug break)
    type: bool  default: --no-print-break-location
--logfile (Specify the name of the log file, use '-' for console, '+' for a temporary file.)
    type: string default: --logfile="v8.log"
--logfile-per-isolate (Separate log files for each isolate.)
    type: bool  default: --logfile-per-isolate
--log (Minimal logging (no API, code, GC, suspect, or handles samples).)
    type: bool  default: --no-log
--log-all (Log all events to the log file.)
    type: bool  default: --no-log-all
--log-source-code (Log source code.)
    type: bool  default: --no-log-source-code
--log-source-position (Log detailed source information.)
    type: bool  default: --no-log-source-position
--log-code (Log code events to the log file without profiling.)
    type: bool  default: --no-log-code
--log-feedback-vector (Log FeedbackVectors on first creation)
    type: bool  default: --no-log-feedback-vector
--log-code-disassemble (Log all disassembled code to the log file.)
    type: bool  default: --no-log-code-disassemble
--log-function-events (Log function events (parse, compile, execute) separately.)
    type: bool  default: --no-log-function-events
--detailed-line-info (Always generate detailed line information for CPU profiling.)
    type: bool  default: --no-detailed-line-info
--prof-sampling-interval (Interval for --prof samples (in microseconds).)
    type: int    default: --prof-sampling-interval=1000
--prof-cpp (Like --prof, but ignore generated code.)
    type: bool  default: --no-prof-cpp
--prof-browser-mode (Used with --prof, turns on browser-compatible mode for profiling.)
    type: bool  default: --prof-browser-mode
--prof (Log statistical profiling information (implies --log-code).)
    type: bool  default: --no-prof
--ll-prof (Enable low-level linux profiler.)
    type: bool  default: --no-ll-prof
--perf-basic-prof (Enable perf linux profiler (basic support).)
    type: bool  default: --no-perf-basic-prof
--perf-basic-prof-only-functions (Only report function code ranges to perf (i.e. no stubs).)
    type: bool  default: --no-perf-basic-prof-only-functions
--perf-prof (Enable perf linux profiler (experimental annotate support).)
    type: bool  default: --no-perf-prof
```

```
--perf-prof-annotate-wasm (Used with --perf-prof, load wasm source map and provide annotate support
(experimental).)
    type: bool default: --no-perf-prof-annotate-wasm
--perf-prof-delete-file (Remove the perf file right after creating it (for testing only).)
    type: bool default: --no-perf-prof-delete-file
--perf-prof-unwinding-info (Enable unwinding info for perf linux profiler (experimental).)
    type: bool default: --no-perf-prof-unwinding-info
--gc-fake-mmap (Specify the name of the file for fake gc mmap used in ll_prof)
    type: string default: --gc-fake-mmap="/tmp/__v8_gc__"
--log-internal-timer-events (Time internal events.)
    type: bool default: --no-log-internal-timer-events
--redirect-code-traces (output deopt information and disassembly into file code-<pid>-<isolate id>.asm)
    type: bool default: --no-redirect-code-traces
--redirect-code-traces-to (output deopt information and disassembly into the given file)
    type: string default: --redirect-code-traces-to=""
--print-opt-source (print source code of optimized and inlined functions)
    type: bool default: --no-print-opt-source
--vtune-prof-annotate-wasm (Used when v8_enable_vtunejit is enabled, load wasm source map and provide annotate
support (experimental).)
    type: bool default: --no-vtune-prof-annotate-wasm
--win64-unwinding-info (Enable unwinding info for Windows/x64)
    type: bool default: --win64-unwinding-info
--interpreted-frames-native-stack (Show interpreted frames on the native stack (useful for external
profilers).)
    type: bool default: --no-interpreted-frames-native-stack
--enable-etw-stack-walking (Enable etw stack walking for windows)
    type: bool default: --no-enable-etw-stack-walking
--trace-elements-transitions (trace elements transitions)
    type: bool default: --no-trace-elements-transitions
--trace-creation-allocation-sites (trace the creation of allocation sites)
    type: bool default: --no-trace-creation-allocation-sites
--print-code (print generated code)
    type: bool default: --no-print-code
--print-opt-code (print optimized code)
    type: bool default: --no-print-opt-code
--print-opt-code-filter (filter for printing optimized code)
    type: string default: --print-opt-code-filter=""
--print-code-verbose (print more information for code)
    type: bool default: --no-print-code-verbose
--print-builtin-code (print generated code for builtins)
    type: bool default: --no-print-builtin-code
--print-builtin-code-filter (filter for printing builtin code)
    type: string default: --print-builtin-code-filter=""
--print-regexp-code (print generated regexp code)
    type: bool default: --no-print-regexp-code
--print-regexp-bytecode (print generated regexp bytecode)
    type: bool default: --no-print-regexp-bytecode
--print-builtin-size (print code size for builtins)
    type: bool default: --no-print-builtin-size
--print-all-code (enable all flags related to printing code)
    type: bool default: --no-print-all-code
--predictable (enable predictable mode)
    type: bool default: --no-predictable
--predictable-gc-schedule (Predictable garbage collection schedule. Fixes heap growing, idle, and memory
reducing behavior.)
    type: bool default: --no-predictable-gc-schedule
--single-threaded (disable the use of background tasks)
    type: bool default: --no-single-threaded
--single-threaded-gc (disable the use of background gc tasks)
    type: bool default: --no-single-threaded-gc
--verify-predictable (this mode is used for checking that V8 behaves predictably)
    type: bool default: --no-verify-predictable
--dump-allocations-digest-at-alloc (dump allocations digest each n-th allocation)
    type: int default: --dump-allocations-digest-at-alloc=-1
```