

**NAME**

run.js – Test-Anything-Protocol module for Node.js

**DESCRIPTION****Usage:**

tap [options] <files>

Executes all the files and interprets their output as TAP formatted test result data.

To parse TAP data from stdin, specify "-" as a filename.

Short options are parsed gnu-style, so for example '**-bCRspec**' would be equivalent to '**--bail --no-color --reporter=spec**'

If the **--check-coverage** or **--coverage-report** options are provided, but no test files are specified, then a coverage report or coverage check will be run on the data from the last test run.

Coverage is never enabled for stdin.

**OPTIONS**

**-j<n> --jobs=<n>**

Run up to <n> test files in parallel Note that this causes tests to be run in "buffered" mode, so line-by-line results cannot be reported, and older TAP parsers may get upset.

**-J --jobs-auto**

Run test files in parallel (auto calculated) Note that this causes tests to be run in "buffered" mode, so line-by-line results cannot be reported, and older TAP parsers may get upset.

**-g<pattern>**

Only run subtests tests matching the specified

**--grep=<pattern>**

pattern.

Patterns are matched against top-level

subtests in each file. To filter tests at subsequent levels, specify this option multiple times.

To specify regular expression flags,

format pattern like a JavaScript RegExp literal. For example: '/xyz/i' for case-insensitive matching.

**-i --invert**

Invert the matches to **--grep** patterns. (Like **grep -v**)

**-c --color**

Use colors (Default for TTY)

**-C --no-color**

Do not use colors (Default for non-TTY)

**-b --bail**

Bail out on first failure

**-B --no-bail**

Do not bail out on first failure (Default)

**-O --only**

Only run tests with {only: true} option

**-R<type> --reporter=<type>**

Use the specified reporter. Defaults to 'classic' when colors are in use, or 'tap' when colors are disabled.

Available reporters:

classic doc dot dump json jsonstream landing list markdown min nyan progress silent spec tap xunit

**-o<file>**

Send the raw TAP output to the specified

**--output-file=<file>**

file. Reporter output will still be printed to stdout, but the file will contain the raw TAP for later reply or analysis.

**-s<file> --save=<file>**

If <file> exists, then it should be a linedelimited list of test files to run. If <file> is not present, then all command-line positional arguments are run.

After the set of test files are run, any

failed test files are written back to the save file.

This way, repeated runs with **-s<file>** will

re-run failures until all the failures are passing, and then once again run all tests.

It's a good idea to .gitignore the file

used for this purpose, as it will churn a lot.

**--coverage --cov**

Capture coverage information using 'nyc'

If a COVERALLS\_REPO\_TOKEN environment

variable is set, then coverage is captured by default and sent to the coveralls.io service.

**--no-coverage --no-cov**

Do not capture coverage information. Note that if nyc is already loaded, then the coverage info will still be captured.

**--coverage-report=<type>**

Output coverage information using the specified istanbul/nyc reporter type.

Default is 'text' when running on the

command line, or 'text-lcov' when piping to coveralls.

If 'html' is used, then the report will

be opened in a web browser after running.

This can be run on its own at any time

after a test run that included coverage.

**--no-coverage-report**

Do not output a coverage report.

**--no-browser**

Do not open a web browser after generating an html coverage report.

**-t<n> --timeout=<n>**

Time out test files after <n> seconds. Defaults to 30, or the value of the TAP\_TIMEOUT environment variable. Setting to 0 allows tests to run forever.

**-T --no-timeout**

Do not time out tests. Equivalent to **--timeout=0**

**-h --help**

print this thing you're looking at

**-v --version**

show the version of this program

**--node-arg=<arg>**

Pass an argument to Node binary in all child processes. Run 'node **--help**' to see a list of all relevant arguments. This can be specified multiple times to pass multiple args to Node.

**--gc --expose-gc**  
Expose the gc() function to Node tests

**--debug**  
Run JavaScript tests with node **--debug**

**--debug-brk**  
Run JavaScript tests with node **--debug-brk**

**--harmony**  
Enable all Harmony flags in JavaScript tests

**--strict**  
Run JS tests in 'use strict' mode

**--test-arg=<arg>**  
Pass an argument to test files spawned by the tap command line executable. This can be specified multiple times to pass multiple args to test scripts.

**--nyc-arg=<arg>**  
Pass an argument to nyc when running child processes with coverage enabled. This can be specified multiple times to pass multiple args to nyc.

**--check-coverage**  
Check whether coverage is within thresholds provided. Setting this explicitly will default **--coverage** to true.

This can be run on its own any time  
after a test run that included coverage.

**--branches**  
what % of branches must be covered? Setting this will default both **--check-coverage** and **--coverage** to true. [default: 0]

**--functions**  
what % of functions must be covered? Setting this explicitly will default both **--check-coverage** and **--coverage** to true. [default: 0]

**--lines** what % of lines must be covered? Setting this explicitly will default both **--check-coverage** and **--coverage** to true. [default: 90]

**--statements**  
what % of statements must be covered? Setting this explicitly will default both **--check-coverage** and **--coverage** to true. [default: 0]

**--100** Full coverage, 100%. Sets branches, statements, functions, and lines to 100.

**--nyc-help**  
Print nyc usage banner. Useful for viewing options for **--nyc-arg**.

**--nyc-version**  
Print version of nyc used by tap.

**--dump-config**  
Dump the config options in JSON format.

**--** Stop parsing flags, and treat any additional command line arguments as filenames.

Environment Variables:

**TAP\_SNAPSHOT**

Set to '1' to generate snapshot files for 't.matchSnapshot()' assertions.

**TAP\_RCFILE**

A yaml formatted file which can set any of the above options. Defaults to \$HOME/.taprc

**TAP\_TIMEOUT**

Default value for **—timeout** option.

**TAP\_COLORS**

Set to '1' to force color output, or '0' to prevent color output.

**TAP\_BAIL**

Bail out on the first test failure. Used internally when **—bailout** is set.

**TAP**

Set to '1' to force standard TAP output, and suppress any reporters. Used when running child tests so that their output is parseable by the test harness.

**TAP\_DIAG**

Set to '1' to show diagnostics by default for passing tests. Set to '0' to NOT show diagnostics by default for failing tests. If not one of these two values, then diagnostics are printed by default for failing tests, and not for passing tests.

**TAP\_BUFFER**

Set to '1' to run subtests in buffered mode by default.

**TAP\_DEV\_LONGSTACK**

Set to '1' to include node–tap internals in stack traces. By default, these are included only when the current working directory is the tap project itself. Note that node internals are always excluded.

**TAP\_DEV\_SHORTSTACK**

Set to '1' to exclude node–tap internals in stack traces, even if the current working directory is the tap project itself.

**\_TAP\_COVERAGE\_**

Reserved for internal use.

**TAP\_DEBUG**

Set to '1' to turn on debug mode.

**NODE\_DEBUG**

Include 'tap' to turn on debug mode.

**TAP\_GREP**

A '\n'–delimited list of grep patterns to apply to root level test objects. (This is an implementation detail for how the **—grep** option works.)

**TAP\_GREP\_INVERT**

Set to '1' to invert the meaning of the patterns in TAP\_GREP. (Implementation detail for how the **—invert** flag works.)

**Config Files:**

You can create a yaml file with any of the options above. By default, the file at ~/.taprc will be loaded, but the TAP\_RCFILE environment variable can modify this.

Run **'tap --dump-config'** for a listing of what can be set in that file. Each of the keys corresponds to one of the options above.