CLI • COMMANDS

sv check

`sv check` finds errors and warnings in your project, such as:

- unused CSS
- accessibility hints
- JavaScript/TypeScript compiler errors

Requires Node 16 or later.

Installation

You will need to have the `svelte-check` package installed in your project:

```
npm i -D svelte-check
```

Usage

```
npx sv check
```

Options

`ignore` are checked.

--output <format>

How to display errors and warnings. See machine-readable output.

- `human`
- `human-verbose`
- `machine`
- `machine-verbose`

--watch

Keeps the process alive and watches for changes.

--preserveWatchOutput

Prevents the screen from being cleared in watch mode.

--tsconfig <path>

Pass a path to a `tsconfig` or `jsconfig` file. The path can be relative to the workspace path or absolute. Doing this means that only files matched by the `files` / `include` / `exclude` pattern of the config file are diagnosed. It also means that errors from TypeScript and JavaScript files are reported. If not given, will traverse upwards from the project directory looking for the next `jsconfig` / `tsconfig.json` file.

--no-tsconfig

Use this if you only want to check the Svelte files found in the current directory and below and ignore any `.js` / `.ts` files (they will not be type-checked)

Files/folders to ignore, relative to workspace root. Paths should be comma-separated and quoted. Example:

```
npx sv check --ignore "dist,build"
```

Only has an effect when used in conjunction with `--no-tsconfig`. When used in conjunction with `--tsconfig`, this will only have effect on the files watched, not on the files that are diagnosed, which is then determined by the `tsconfig.json`.

--fail-on-warnings

If provided, warnings will cause `sv check` to exit with an error code.

--compiler-warnings <warnings>

A quoted, comma-separated list of `code:behaviour` pairs where `code` is a compiler warning code and `behaviour` is either `ignore` or `error`:

```
npx sv check --compiler-warnings "css_unused_selector:ignore,a11y_missing_attribute:e
```

--diagnostic-sources <sources>

A quoted, comma-separated list of sources that should run diagnostics on your code. By default, all are active:

- `js` (includes TypeScript)
- `svelte`
- `css`

Example:

Filters the diagnostics:

- `warning` (default) — both errors and warnings are shown
- `error` — only errors are shown

Troubleshooting

See the language-tools documentation for more information on preprocessor setup and other troubleshooting.

Machine-readable output

Setting the `--output` to `machine` or `machine-verbose` will format output in a way that is easier to read by machines, e.g. inside CI pipelines, for code quality checks, etc.

Each row corresponds to a new record. Rows are made up of columns that are separated by a single space character. The first column of every row contains a timestamp in milliseconds which can be used for monitoring purposes. The second column gives us the "row type", based on which the number and types of subsequent columns may differ.

The first row is of type `START` and contains the workspace folder (wrapped in quotes). Example:

```
1590680325583 START "/home/user/language-tools/packages/language-server/test/plugins/type
```

Any number of `ERROR` or `WARNING` records may follow. Their structure is identical and depends on the output argoument.

If the argument is `machine` it will tell us the filename, the starting line and column numbers, and the error message. The filename is relative to the workspace directory. The

```
1590680326778 WARNING  imported-file.svelte  0:37  Component has unused export property  |
```

If the argument is `machine-verbose` it will tell us the filename, the starting line and column numbers, the ending line and column numbers, the error message, the code of diagnostic, the human-friendly description of the code and the human-friendly source of the diagnostic (eg. svelte/typescript). The filename is relative to the workspace directory. Each diagnostic is represented as an ndjson line prefixed by the timestamp of the log. Example:

```
1590680326283 {"type":"ERROR","fn":"codeaction.svelte","start":{"line":1,"character":16},
1590680326778 {"type":"WARNING","filename":"imported-file.svelte","start":{"line":0,"chara
const prop`","code":"unused-export-let","source":"svelte"}
```

The output concludes with a `COMPLETED` message that summarizes total numbers of files, errors and warnings that were encountered during the check. Example:

```
1590680326807 COMPLETED 20 FILES 21 ERRORS 1 WARNINGS 3 FILES_WITH_PROBLEMS
```

If the application experiences a runtime error, this error will appear as a `FAILURE` record. Example:

```
1590680328921 FAILURE "Connection closed"
```

Credits

- Vue's VTI which laid the foundation for `svelte-check`

FAQ

a component prop but didn't update any of the places where the prop is used — the usage sites are all errors now, but you would miss them if checks only ran on changed files.

Edit this page on GitHub

PREVIOUS
sv add

NEXT
sv migrate