tokei Count lines of code

CONCEPTS

Each language row splits lines into code, comments and blanks. Embedded code (CSS in HTML, say) is counted under its own language as a child row.

tokei honours .gitignore, .ignore and .tokeignore by default, so build output and vendored dirs listed there are not counted.

COUNTING

`tokei`
Table for the current directory

`tokei <dir> <dir>`
Count several paths together

`tokei -f`
Show every file

`tokei -s <key>`
Sort rows by files, lines, blanks, code or comments

`tokei -s code -f`
Biggest files by code lines

`tokei --hidden`
Include hidden files

`tokei -c <n>`
Fixed output width in columns

`tokei -l`
List supported languages and extensions

`tokei -v`
Log unknown extensions (-vv for more)

`tokei -V`
Print the version

FILTERING

`tokei -t=<lang>,<lang>`
Only these languages

`tokei -t=Rust,Markdown`
Example: Rust and Markdown only

`tokei -e <pattern>`
Exclude files or dirs, gitignore syntax

`tokei -e '*.d' -e <dir>`
Repeat -e for several patterns

`tokei --no-ignore`
Count everything, ignore all ignore files

`tokei --no-ignore-vcs`
Skip only .gitignore/.hgignore

`tokei --no-ignore-dot`
Skip .ignore and .tokeignore

`tokei --no-ignore-parent`
Skip ignore files in parent dirs

Quote glob patterns so the shell does not expand them first.

OUTPUT AND CONFIG

`tokei -o json`
Print JSON instead of a table

`tokei -o yaml`
Print YAML

`tokei -o cbor`
Print CBOR (binary)

`tokei -o json > stats.json`
Save a run for later

`tokei -i stats.json`
Load a saved run instead of scanning

OUTPUT AND CONFIG (cont.)

`tokei -i stdin`
Read a saved run from a pipe

Some builds lack json, yaml or cbor. Reinstall with cargo install tokei --features all.

`.tokeignore`
Per-project exclude list, gitignore syntax

`tokei.toml / .tokeirc`
Default options; read from cwd, home, config dir

Install

Where	Command
`Cargo`	cargo install tokei
`Homebrew`	brew install tokei
`Arch`	pacman -S tokei
`Fedora`	sudo dnf install tokei
`Windows`	winget install XAMPPRocky.tokei

RECIPES

Source only, biggest files first

```
tokei -f -s code -t=Rust src
```

Compare two points in history

```
tokei -o json > now.json
git stash -q && git checkout -q
<rev>
tokei -o json > then.json
git checkout -q - && git stash pop
-q
```

Total code lines in a script

```
tokei -o json | jq '.Total.code'
```

Ignore tests and vendored code

```
printf 'vendor/\n*_test.go\n'
>> .tokeignore
tokei
```