CLI

`info`

npm Yarn pnpm **Cargo**

```
$ cargo tauri info
```

```
Usage: cargo-tauri info [OPTIONS]

Options:
      --interactive  Interactive mode to apply automatic fixes
  -v, --verbose...   Enables verbose logging
  -h, --help         Print help
  -V, --version      Print version
```

It shows a concise list of information about the environment, Rust, Node.js and their versions as well as some relevant configurations.

> **INFO**
>
> This command is pretty helpful when you need to have a quick overview of your application. When requesting some help, it can be useful that you share this report with us.

`init`

npm Yarn pnpm **Cargo**

```
$ cargo tauri init
```

```
Usage: cargo-tauri init [OPTIONS]

Options:
      --ci
```

```
          Skip prompting for values
  -v, --verbose...
          Enables verbose logging
  -f, --force
          Force init to overwrite the src-tauri folder
  -l, --log
          Enables logging
  -d, --directory <DIRECTORY>
          Set target directory for init [default: /home/runner/work/tauri-docs/tauri-docs]
  -t, --tauri-path <TAURI_PATH>
          Path of the Tauri project to use (relative to the cwd)
  -A, --app-name <APP_NAME>
          Name of your Tauri application
  -W, --window-title <WINDOW_TITLE>
          Window title of your Tauri application
  -D, --dist-dir <DIST_DIR>
          Web assets location, relative to <project-dir>/src-tauri
  -P, --dev-path <DEV_PATH>
          Url of your dev server
      --before-dev-command <BEFORE_DEV_COMMAND>
          A shell command to run before `tauri dev` kicks in
      --before-build-command <BEFORE_BUILD_COMMAND>
          A shell command to run before `tauri build` kicks in
  -h, --help
          Print help
  -V, --version
          Print version
```

`plugin init`

npm Yarn pnpm **Cargo**

```
$ cargo tauri plugin init
```

```
Usage: cargo-tauri plugin init [OPTIONS] --name <PLUGIN_NAME>

Options:
  -n, --name <PLUGIN_NAME>       Name of your Tauri plugin
  -v, --verbose...               Enables verbose logging
      --api                      Initializes a Tauri plugin with TypeScript API
  -d, --directory <DIRECTORY>    Set target directory for init [default:
/home/runner/work/tauri-docs/tauri-docs]
  -t, --tauri-path <TAURI_PATH>  Path of the Tauri project to use (relative to the cwd)
  -a, --author <AUTHOR>          Author name
  -h, --help                      Print help
  -V, --version                   Print version
```

dev

npm Yarn pnpm **Cargo**

```
$ cargo tauri dev
```

```
Usage: cargo-tauri dev [OPTIONS] [ARGS]...

Arguments:
  [ARGS]...  Command line arguments passed to the runner. Arguments after `--` are passed to
the application

Options:
  -r, --runner <RUNNER>            Binary to use to run the application
  -v, --verbose...                 Enables verbose logging
  -t, --target <TARGET>            Target triple to build against
  -f, --features [<FEATURES>...]   List of cargo features to activate
  -e, --exit-on-panic              Exit on panic
  -c, --config <CONFIG>            JSON string or path to JSON file to merge with
tauri.conf.json
      --release                    Run the code in release mode
      --no-watch                   Disable the file watcher
      --no-dev-server              Disable the dev server for static files
      --port <PORT>                Specify port for the dev server for static files. Defaults to
1430 Can also be set using `TAURI_DEV_SERVER_PORT` env var
  -h, --help                       Print help
  -V, --version                    Print version
```

This command will open the WebView in development mode. It makes use of the `build.devPath` property from your `src-tauri/tauri.conf.json` file.

If you have entered a command to the `build.beforeDevCommand` property, this one will be executed before the `dev` command.

See more about the configuration.

TROUBLESHOOTING

If you're not using `build.beforeDevCommand`, make sure your `build.devPath` is correct and, if using a development server, that it's started before using this command.

`build`

npm Yarn pnpm **Cargo**

```
$ cargo tauri build
```

```
Usage: cargo-tauri build [OPTIONS] [ARGS]...

Arguments:
  [ARGS]...
          Command line arguments passed to the runner

Options:
  -r, --runner <RUNNER>
          Binary to use to build the application, defaults to `cargo`

  -v, --verbose...
          Enables verbose logging

  -d, --debug
          Builds with the debug flag

  -t, --target <TARGET>
          Target triple to build against.

          It must be one of the values outputted by `$rustc --print target-list` or `universal-apple-darwin` for an universal macOS application.

          Note that compiling an universal macOS application requires both `aarch64-apple-darwin` and `x86_64-apple-darwin` targets to be installed.

  -f, --features [<FEATURES>...]
          Space or comma separated list of features to activate

  -b, --bundles [<BUNDLES>...]
          Space or comma separated list of bundles to package.

          Each bundle must be one of `deb`, `appimage`, `msi`, `app` or `dmg` on MacOS and `updater` on all platforms. If `none` is specified, the bundler will be skipped.
          Note that the `updater` bundle is not automatically added so you must specify it if
the updater is enabled.

  -c, --config <CONFIG>
          JSON string or path to JSON file to merge with tauri.conf.json

      --ci
          Skip prompting for values

  -h, --help
          Print help (see a summary with '-h')

  -V, --version
          Print version
```

This command will bundle your application, either in production mode or debug mode if you used the `--debug` flag. It makes use of the `build.distDir` property from your `src-tauri/tauri.conf.json` file.

If you have entered a command to the `build.beforeBuildCommand` property, this one will be executed before the `build` command.

See more about the configuration.

`icon`

npm Yarn pnpm Cargo

```
$ cargo tauri icon
```

```
Usage: cargo-tauri icon [OPTIONS] [INPUT]

Arguments:
  [INPUT]  Path to the source icon (png, 1024x1024px with transparency) [default: ./app-
icon.png]

Options:
  -o, --output <OUTPUT>  Output directory. Default: 'icons' directory next to the
tauri.conf.json file
  -v, --verbose...       Enables verbose logging
  -p, --png <PNG>        Custom PNG icon sizes to generate. When set, the default icons are not
generated
```

```
  -h, --help             Print help
  -V, --version          Print version
```

For more information, check out the complete Tauri Icon Guide.

`completions`

npm Yarn pnpm **Cargo**

```
$ cargo tauri completions
```

```
Usage: cargo-tauri completions [OPTIONS] --shell <SHELL>

Options:
  -s, --shell <SHELL>    Shell to generate a completion script for. [possible values: bash,
elvish, fish, powershell, zsh]
  -v, --verbose...       Enables verbose logging
  -o, --output <OUTPUT>  Output file for the shell completions. By default the completions are
printed to stdout
  -h, --help             Print help
  -V, --version          Print version
```

The Tauri CLI can generate shell completions for Bash, Zsh, PowerShell and Fish.

Here are some instructions to configure Bash, Zsh and PowerShell. If you face an issue, please follow your shell's instructions instead. Note that it is recommended to check the generated completions script before executing it for security reasons.

Bash

Get the Bash completions and move to a known folder:

npm Yarn pnpm **Cargo**

```
$ cargo tauri completions --shell bash > tauri.sh
$ mv tauri.sh /usr/local/etc/bash_completion.d/tauri.bash
```

Load the completions script by adding the following to `.bashrc`:

```
$ source /usr/local/etc/bash_completion.d/tauri.bash
```

Zsh

Get the Zsh completions and move to a known folder:

npm Yarn pnpm **Cargo**

```
$ cargo tauri completions --shell zsh > completions.zsh
$ mv completions.zsh $HOME/.completions/_tauri
```

Load the completions folder using fpath adding the following to `.zshrc`:

```
$ fpath=(~/.completions $fpath)
$ autoload -U compinit
```

PowerShell

Get the PowerShell completions and add it to the `$profile` file to execute it on all sessions:

npm Yarn pnpm **Cargo**

```
PS C:\> cargo tauri completions --shell powershell > ((Split-Path -Path
$profile)+"\_tauri.ps1")
PS C:\> Add-Content -Path $profile -Value '& "$PSScriptRoot\_tauri.ps1"'
```

`version`

npm Yarn pnpm **Cargo**

```
$ cargo tauri --version
```

```
  Description
    Returns the current version of tauri
```

This command will show the current version of Tauri.

CLI usage

See more about the usage through this complete guide.

Edit this page

*Last updated on **Sep 30, 2023***