Crate tauri

Tauri is a framework for building tiny, blazing fast binaries for all major desktop platforms. Developers can integrate any front-end framework that compiles to HTML, JS and CSS for building their user interface. The backend of the application is a rust-sourced binary with an API that the front-end can interact with.

Cargo features

The following are a list of Cargo features that can be enabled or disabled:

- **wry** *(enabled by default)*: Enables the wry runtime. Only disable it if you want a custom runtime.
- **test**: Enables the [`test`] module exposing unit test helpers.
- **dox**: Internal feature to generate Rust documentation without linking on Linux.
- **objc-exception**: Wrap each msg_send! in a @try/@catch and panics if an exception is caught, preventing Objective-C from unwinding into Rust.
- **linux-protocol-headers**: Enables headers support for custom protocol requests on Linux. Requires webkit2gtk v2.36 or above.
- **isolation**: Enables the isolation pattern. Enabled by default if the `tauri > pattern > use` config option is set to `isolation` on the `tauri.conf.json` file.
- **custom-protocol**: Feature managed by the Tauri CLI. When enabled, Tauri assumes a production environment instead of a development one.
- **updater**: Enables the application auto updater. Enabled by default if the `updater` config is defined on the `tauri.conf.json` file.
- **devtools**: Enables the developer tools (Web inspector) and `Window::open_devtools`. Enabled by default on debug builds. On macOS it uses private APIs, so you can't enable it if your app will be published to the App Store.
- **shell-open-api**: Enables the `api::shell` module.
- **http-api**: Enables the `api::http` module.
- **http-multipart**: Adds support to `multipart/form-data` requests.
- **reqwest-client**: Alias for the `http-api` feature flag.
- **native-tls-vendored**: Compile and statically link to a vendored copy of OpenSSL.
- **reqwest-native-tls-vendored**: Alias for the `native-tls-vendored` feature flag.
- **os-api**: Enables the `api::os` module.
- **process-command-api**: Enables the `api::process::Command` APIs.
- **global-shortcut**: Enables the global shortcut APIs.
- **clipboard**: Enables the clipboard APIs.
- **process-relaunch-dangerous-allow-symlink-macos**: Allows the `api::process::current_binary` function to allow symlinks on macOS (this is dangerous, see the Security section in the documentation website).
- **dialog**: Enables the `api::dialog` module.
- **notification**: Enables the `api::notification` module.
- **fs-extract-api**: Enabled the `tauri::api::file::Extract` API.

- **cli**: Enables usage of `clap` for CLI argument parsing. Enabled by default if the `cli` config is defined on the `tauri.conf.json` file.
- **system-tray**: Enables application system tray API. Enabled by default if the `systemTray` config is defined on the `tauri.conf.json` file.
- **macos-private-api**: Enables features only available in **macOS**'s private APIs, currently the `transparent` window functionality and the `fullScreenEnabled` preference setting to `true`. Enabled by default if the `tauri > macosPrivateApi` config flag is set to `true` on the `tauri.conf.json` file.
- **windows7-compat**: Enables compatibility with Windows 7 for the notification API.
- **window-data-url**: Enables usage of data URLs on the webview.
- **compression** *(enabled by default): Enables asset compression. You should only disable this if you want faster compile times in release builds - it produces larger binaries.
- **config-json5**: Adds support to JSON5 format for `tauri.conf.json`.
- **config-toml**: Adds support to TOML format for the configuration `Tauri.toml`.
- **icon-ico**: Adds support to set `.ico` window icons. Enables `Icon::File` and `Icon::Raw` variants.
- **icon-png**: Adds support to set `.png` window icons. Enables `Icon::File` and `Icon::Raw` variants.

Cargo allowlist features

The following are a list of Cargo features that enables commands for Tauri's API package. These features are automatically enabled by the Tauri CLI based on the `allowlist` configuration under `tauri.conf.json`.

- **api-all**: Enables all API endpoints.

Clipboard allowlist

- **clipboard-all**: Enables all Clipboard APIs.
- **clipboard-read-text**: Enables the `readText` API.
- **clipboard-write-text**: Enables the `writeText` API.

Dialog allowlist

- **dialog-all**: Enables all Dialog APIs.
- **dialog-ask**: Enables the `ask` API.
- **dialog-confirm**: Enables the `confirm` API.
- **dialog-message**: Enables the `message` API.
- **dialog-open**: Enables the `open` API.
- **dialog-save**: Enables the `save` API.

Filesystem allowlist

- **fs-all**: Enables all Filesystem APIs.
- **fs-copy-file**: Enables the `copyFile` API.
- **fs-create-dir**: Enables the `createDir` API.
- **fs-exists**: Enables the `exists` API.
- **fs-read-dir**: Enables the `readDir` API.

- **fs-read-file**: Enables the `readTextFile` API and the `readBinaryFile` API.
- **fs-remove-dir**: Enables the `removeDir` API.
- **fs-remove-file**: Enables the `removeFile` API.
- **fs-rename-file**: Enables the `renameFile` API.
- **fs-write-file**: Enables the `writeFile` API and the `writeBinaryFile` API.

Global shortcut allowlist

- **global-shortcut-all**: Enables all GlobalShortcut APIs.

HTTP allowlist

- **http-all**: Enables all HTTP APIs.
- **http-request**: Enables the `request` APIs.

Notification allowlist

- **notification-all**: Enables all Notification APIs.

OS allowlist

- **os-all**: Enables all OS APIs.

Path allowlist

- **path-all**: Enables all Path APIs.

Process allowlist

- **process-all**: Enables all Process APIs.
- **process-exit**: Enables the `exit` API.
- **process-relaunch**: Enables the `relaunch` API.

Protocol allowlist

- **protocol-all**: Enables all Protocol APIs.
- **protocol-asset**: Enables the `asset` custom protocol.

Shell allowlist

- **shell-all**: Enables all Clipboard APIs.
- **shell-execute**: Enables executing arbitrary programs.
- **shell-sidecar**: Enables executing a `sidecar` program.
- **shell-open**: Enables the `open` API.

Window allowlist

- **window-all**: Enables all Window APIs.
- **window-create**: Enables the API used to create new windows.
- **window-center**: Enables the `center` API.
- **window-request-user-attention**: Enables the `requestUserAttention` API.
- **window-set-resizable**: Enables the `setResizable` API.
- **window-set-maximizable**: Enables the `setMaximizable` API.
- **window-set-minimizable**: Enables the `setMinimizable` API.
- **window-set-closable**: Enables the `setClosable` API.
- **window-set-title**: Enables the `setTitle` API.
- **window-maximize**: Enables the `maximize` API.
- **window-unmaximize**: Enables the `unmaximize` API.
- **window-minimize**: Enables the `minimize` API.
- **window-unminimize**: Enables the `unminimize` API.
- **window-show**: Enables the `show` API.
- **window-hide**: Enables the `hide` API.
- **window-close**: Enables the `close` API.
- **window-set-decorations**: Enables the `setDecorations` API.
- **window-set-always-on-top**: Enables the `setAlwaysOnTop` API.
- **window-set-content-protected**: Enables the `setContentProtected` API.
- **window-set-size**: Enables the `setSize` API.
- **window-set-min-size**: Enables the `setMinSize` API.
- **window-set-max-size**: Enables the `setMaxSize` API.
- **window-set-position**: Enables the `setPosition` API.
- **window-set-fullscreen**: Enables the `setFullscreen` API.
- **window-set-focus**: Enables the `setFocus` API.
- **window-set-icon**: Enables the `setIcon` API.
- **window-set-skip-taskbar**: Enables the `setSkipTaskbar` API.
- **window-set-cursor-grab**: Enables the `setCursorGrab` API.
- **window-set-cursor-visible**: Enables the `setCursorVisible` API.
- **window-set-cursor-icon**: Enables the `setCursorIcon` API.
- **window-set-cursor-position**: Enables the `setCursorPosition` API.
- **window-set-ignore-cursor-events**: Enables the `setIgnoreCursorEvents` API.
- **window-start-dragging**: Enables the `startDragging` API.
- **window-print**: Enables the `print` API.

App allowlist

- **app-all**: Enables all App APIs.
- **app-show**: Enables the `show` API.
- **app-hide**: Enables the `hide` API.

Re-exports

```
pub use self::window::Monitor;
pub use self::window::Window;
pub use self::window::WindowBuilder;
pub use tauri_utils as utils;
pub use scope::*;
```

Modules

api	The Tauri API interface.
async_runtime	The singleton async runtime used by Tauri and exposed to users.
command	The Tauri custom commands types and traits.
http	
plugin	The Tauri plugin extension to expand Tauri functionality.
scope	The allowlist scopes.
test `test`	Utilities for unit testing on Tauri applications.
updater `updater`	The Tauri updater.
window	The Tauri window types and functions.

Macros

generate_context	Reads the config file at compile time and generates a `Context` based on its content.
generate_handler	Accepts a list of commands functions. Creates a handler that allows commands to be called from JS with invoke().
tauri_build_context	Include a `Context` that was generated by `tauri-build` inside your build script.

Structs

AboutMetadata	Application metadata for the `MenuItem::About` action.
App	The instance of the currently running application.
AppHandle	A handle to the currently running application.
Asset	A resolved asset.
AssetResolver	The asset resolver is a helper to access the `tauri_utils::assets::Assets` interface.
Builder	Builds a Tauri application.
CloseRequestApi	Api exposed on the `CloseRequested` event.
Config	The Tauri configuration object. It is read from a file where you can define your frontend assets, configure the bundler, enable the app updater, define a system tray, enable APIs via the allowlist and more.
Context	User supplied data required inside of a Tauri application.
CustomMenuItem	A custom menu item.
Env	Information about environment variables.
Event	An event that was triggered.

EventHandler	Represents an event handler.
GlobalWindowEvent	A window event that was triggered on the specified window.
Invoke	The message and resolver given to a custom command.
InvokeError	Error response from an `InvokeMessage`.
InvokeMessage	An invoke message.
InvokePayload	The payload used on the IPC invoke.
InvokeResolver	Resolver of a invoke message.
LogicalPosition	A position represented in logical pixels.
LogicalSize	A size represented in logical pixels.
Menu	A window menu.
MenuEvent	The window menu event.
PackageInfo	`tauri::App` package information.
PageLoadPayload	The payload for the `OnPageLoad` hook.
PathResolver	The path resolver is a helper for the application-specific `crate::api::path` APIs.
PhysicalPosition	A position represented in physical pixels.
PhysicalSize	A size represented in physical pixels.
RunIteration	Metadata for a runtime event loop iteration on `run_iteration`.
State	A guard for a state value.
StateManager	The Tauri state manager.
Submenu	
SystemTray `system-tray`	Represents a System Tray instance.
SystemTrayHandle `system-tray`	A handle to a system tray. Allows updating the context menu items.
SystemTrayMenu `system-tray`	A system tray menu.
SystemTrayMenuItemHandle `system-tray`	A handle to a system tray menu item.
SystemTraySubmenu `system-tray`	
WebviewAttributes	The attributes used to create an webview.
WindowMenuEvent	A menu event that was triggered on a window.

Enums

CursorIcon	Describes the appearance of the mouse cursor.
DeviceEventFilter	
Error	The Tauri error enum. Runtime errors that can happen inside a Tauri application.
EventLoopMessage	The user event type.
FileDropEvent	The file drop event payload.
Icon	A icon definition.
InvokeResponse	Response from a `InvokeMessage` passed to the `InvokeResolver`.
MenuEntry	An entry on the system tray menu.

MenuItem	A menu item, bound to a pre-defined action or `Custom` emit an event. Note that status bar only supports `Custom` menu item variants. And on the menu bar, some platforms might not support some of the variants. Unsupported variant will be no-op on such platform.
Pattern	An application pattern.
Position	A position that's either physical or logical.
RunEvent	An application event, triggered from the event loop.
Size	A size that's either physical or logical.
SystemTrayEvent `system-tray`	System tray event.
SystemTrayMenuItem `system-tray`	System tray menu item.
Theme	System theme.
UpdaterEvent `updater`	Updater events.
UserAttentionType	Type of user attention requested on a window.
WindowEvent	An event from a window.
WindowUrl	An URL to open on a Tauri webview window.

Constants

VERSION	The Tauri version.

Traits

Assets	Represents a container of file assets that are retrievable during runtime.
ClipboardManager `clipboard`	Clipboard manager.
GlobalShortcutManager `global-shortcut`	A global shortcut manager.
Manager	Manages a running application.
Pixel	
Runtime	The webview runtime interface. A wrapper around `runtime::Runtime` with the proper user event type associated.

Functions

webview_version `wry`	Get Webview/Webkit version on current platform.

Type Aliases

InvokeHandler	A closure that is run every time Tauri receives a message it doesn't explicitly handle.
InvokeResponder	A closure that is responsible for respond a JS message.

OnPageLoad	A closure that is run once every time a window is created and loaded.
Result	`Result<T, ::tauri::Error>`
SetupHook	A closure that is run when the Tauri application is setting up.
SyncTask	A task to run on the main thread.
Wry `wry`	A Tauri `Runtime` wrapper around wry.

Attribute Macros

command	Mark a function as a command handler. It creates a wrapper function with the necessary glue code.