Splashscreen

If your webpage could take some time to load, or if you need to run an initialization procedure in Rust before displaying your main window, a splashscreen could improve the loading experience for the user.

Setup

First, create a `splashscreen.html` in your `distDir` that contains the HTML code for a splashscreen. Then, update your `tauri.conf.json` like so:

```
"windows": [
  {
    "title": "Tauri App",
    "width": 800,
    "height": 600,
    "resizable": true,
    "fullscreen": false,
+   "visible": false // Hide the main window by default
  },
  // Add the splashscreen window
+ {
+   "width": 400,
+   "height": 200,
+   "decorations": false,
+   "url": "splashscreen.html",
+   "label": "splashscreen"
+ }
]
```

Now, your main window will be hidden and the splashscreen window will show when your app is launched. Next, you'll need a way to close the splashscreen and show the main window when your app is ready. How you do this depends on what you are waiting for before closing the splashscreen.

Waiting for Webpage

If you are waiting for your web code, you'll want to create a `close_splashscreen` command.

```
use tauri::{Manager, Window};
// Create the command:
```

```
// This command must be async so that it doesn't run on the main thread.
#[tauri::command]
async fn close_splashscreen(window: Window) {
  // Close splashscreen
  window.get_window("splashscreen").expect("no window labeled 'splashscreen'
found").close().unwrap()
  // Show main window
  window.get_window("main").expect("no window labeled 'main' found").show().unwrap();
}

// Register the command:
fn main() {
  tauri::Builder::default()
    // Add this line
    .invoke_handler(tauri::generate_handler![close_splashscreen])
    .run(tauri::generate_context!())
    .expect("failed to run app");
}
```

You can then import it to your project in one of two ways:

```
// With the Tauri API npm package:
import { invoke } from '@tauri-apps/api/tauri'
```

or

```
// With the Tauri global script:
const invoke = window.__TAURI__.invoke
```

And finally, add an Event Listener (or just call `invoke()` whenever you want):

```
document.addEventListener('DOMContentLoaded', () => {
  // This will wait for the window to load, but you could
  // run this function on whatever trigger you want
  invoke('close_splashscreen')
})
```

Waiting for Rust

If you are waiting for Rust code to run, put it in the `setup` function handler so you have access to the `App` instance:

```rust
use tauri::Manager;

fn main() {
  tauri::Builder::default()
    .setup(|app| {
      let splashscreen_window = app.get_window("splashscreen").unwrap();
      let main_window = app.get_window("main").unwrap();
      // we perform the initialization code on a new task so the app doesn't freeze
      tauri::async_runtime::spawn(async move {
        // initialize your app here instead of sleeping :)
        println!("Initializing...");
        std::thread::sleep(std::time::Duration::from_secs(2));
        println!("Done initializing.");

        // After it's done, close the splashscreen and display the main window
        splashscreen_window.close().unwrap();
        main_window.show().unwrap();
      });
      Ok(())
    })
    .run(tauri::generate_context!())
    .expect("failed to run app");
}
```