FileSystemModule

August 17, 2026

FileSystemModule

Kind: Interface

Source: `src/helper/ssg/ssg.ts`

Part of: Helper

`FileSystemModule` defines the file-system operations used by the static site generation helper. It abstracts directory creation and file writing so the generator can work with a supplied filesystem implementation.

Diagram

```
graph LR
  SSG[SSG helper] --> FS[FileSystemModule]
  FS --> MKDIR[mkdir()]
  FS --> WRITE[writeFile()]
  MKDIR --> Directory[Output directory]
  WRITE --> File[Generated file]
```

Usage

```
const fileSystem: FileSystemModule = {
  async mkdir(directoryPath) {
    return directoryPath;
  },

  async writeFile(filePath, content) {
    await fs.promises.writeFile(filePath, content);
  },
};

await fileSystem.mkdir("dist");
await fileSystem.writeFile("dist/index.html", "<h1>Hello</h1>");
```

AI Coding Instructions

- Implement `mkdir()` with a return type compatible with `Promise<void | string>`.
- Implement `writeFile()` so it resolves only after the target file has been written.
- Create output directories before writing generated files into them.
- Keep filesystem-specific behavior inside the `FileSystemModule` implementation.

How it works

`FileSystemModule` is an experimental TypeScript interface that abstracts the two filesystem operations required by the static-site-generation (SSG) writer: creating directories and writing files. Its API may change. [src/helper/ssg/ssg.ts:29-37]

It requires these asynchronous methods:

- `writeFile(path, data)`: writes a `string` or `Uint8Array` to `path` and resolves with no value. [src/helper/ssg/ssg.ts:34-36]
- `mkdir(path, { recursive })`: creates a directory, receives a required `recursive` boolean option, and may resolve with `void` or a `string`. [src/helper/ssg/ssg.ts:34-37]

Role in SSG generation

`toSSG` accepts a `FileSystemModule` as its second argument, then passes it to `saveContentToFile` for every generated route response. [src/helper/ssg/ssg.ts:341-347] [src/helper/ssg/ssg.ts:368-369] [src/helper/ssg/ssg.ts:442-445]

Before writing, `saveContentToFile` derives an output filename from the route path, output directory, MIME type, and optional extension map. [src/helper/ssg/ssg.ts:310-315] [src/helper/ssg/ssg.ts:320-322] It rejects a derived path outside the configured output directory by throwing `Error: Path traversal detected: "<path>" is outside the output directory`. [src/helper/ssg/utils.ts:77-86]

For each directory path not already recorded in a module-level `Set`, the writer calls `fsModule.mkdir(dirPath, { recursive: true })` and records that directory after the promise resolves. [src/helper/ssg/ssg.ts:309] [src/helper/ssg/ssg.ts:323-327] The directory cache is shared at module scope, rather than being reset per `toSSG` call. [src/helper/ssg/ssg.ts:309] The code then calls `writeFile` with string content unchanged, or converts an `ArrayBuffer` to `Uint8Array` before writing. [src/helper/ssg/ssg.ts:328-332]

A rejected `mkdir` or `writeFile` call is caught by the SSG generation flow; `toSSG` returns `{ success: false, files: [], error }`, converting a non-`Error` rejection into `new Error(String(error))`. [src/helper/ssg/ssg.ts:443-445] [src/helper/ssg/ssg.ts:451-463]

Hook interaction

`AfterGenerateHook` receives the same `FileSystemModule` instance supplied to `toSSG`, along with the result and optional SSG options. [src/helper/ssg/ssg.ts:110-116] [src/helper/ssg/ssg.ts:160-172] `toSSG` invokes these hooks after forming either its success or failure result. [src/helper/ssg/ssg.ts:460-469]

Included adapter implementations

The Bun adapter exports `bunFileSystemModule`, whose `writeFile` delegates to `Bun.write`; its `mkdir` method is an async no-op. [src/adapter/bun/ssg.ts:5-18] Its adapter-level `toSSG` passes that module to the shared SSG implementation. [src/adapter/bun/ssg.ts:25-27]

The Deno adapter exports `denoFileSystemModule`, whose `writeFile` encodes strings with `TextEncoder` and otherwise wraps the input as `Uint8Array` before calling `Deno.writeFile`; its `mkdir` delegates to `Deno.mkdir` with the passed recursive setting. [src/adapter/deno/ssg.ts:9-18] Its adapter-level `toSSG` likewise passes that module to the shared implementation. [src/adapter/deno/ssg.ts:25-27]