import {RawSourceMap} from 'source-map-js';

import {Options, StringOptions} from './options';

/\*\*

\* The result of compiling Sass to CSS. Returned by [[compile]],

\* [[compileAsync]], [[compileString]], and [[compileStringAsync]].

\*

\* @category Compile

\*/

export interface CompileResult {

/\*\*

\* The generated CSS.

\*

\* Note that this \*never\* includes a `sourceMapUrl` comment—it's up to the

\* caller to determine where to save the source map and how to link to it from

\* the stylesheet.

\*/

css: string;

/\*\*

\* The canonical URLs of all the stylesheets that were loaded during the

\* Sass compilation. The order of these URLs is not guaranteed.

\*/

loadedUrls: URL[];

/\*\*

\* The object representation of the source map that maps locations in the

\* generated CSS back to locations in the Sass source code.

\*

\* This typically uses absolute `file:` URLs to refer to Sass files, although

\* this can be controlled by having a custom [[Importer]] return

\* [[ImporterResult.sourceMapUrl]].

\*

\* This is set if and only if [[Options.sourceMap]] is `true`.

\*/

sourceMap?: RawSourceMap;

}

/\*\*

\* Synchronously compiles the Sass file at `path` to CSS. If it succeeds it

\* returns a [[CompileResult]], and if it fails it throws an [[Exception]].

\*

\* This only allows synchronous [[Importer]]s and [[CustomFunction]]s.

\*

\* @example

\*

\* ```js

\* const sass = require('sass');

\*

\* const result = sass.compile("style.scss");

\* console.log(result.css);

\* ```

\*

\* @category Compile

\* @compatibility dart: "1.45.0", node: false

\*/

export function compile(path: string, options?: Options<'sync'>): CompileResult;

/\*\*

\* Asynchronously compiles the Sass file at `path` to CSS. Returns a promise

\* that resolves with a [[CompileResult]] if it succeeds and rejects with an

\* [[Exception]] if it fails.

\*

\* This only allows synchronous or asynchronous [[Importer]]s and

\* [[CustomFunction]]s.

\*

\* \*\*Heads up!\*\* When using Dart Sass, \*\*[[compile]] is almost twice as fast as

\* [[compileAsync]]\*\*, due to the overhead of making the entire evaluation

\* process asynchronous.

\*

\* @example

\*

\* ```js

\* const sass = require('sass');

\*

\* const result = await sass.compileAsync("style.scss");

\* console.log(result.css);

\* ```

\*

\* @category Compile

\* @compatibility dart: "1.45.0", node: false

\*/

export function compileAsync(

path: string,

options?: Options<'async'>

): Promise<CompileResult>;

/\*\*

\* Synchronously compiles a stylesheet whose contents is `source` to CSS. If it

\* succeeds it returns a [[CompileResult]], and if it fails it throws an

\* [[Exception]].

\*

\* This only allows synchronous [[Importer]]s and [[CustomFunction]]s.

\*

\* @example

\*

\* ```js

\* const sass = require('sass');

\*

\* const result = sass.compileString(`

\* h1 {

\* font-size: 40px;

\* code {

\* font-face: Roboto Mono;

\* }

\* }`);

\* console.log(result.css);

\* ```

\*

\* @category Compile

\* @compatibility dart: "1.45.0", node: false

\*/

export function compileString(

source: string,

options?: StringOptions<'sync'>

): CompileResult;

/\*\*

\* Asynchronously compiles a stylesheet whose contents is `source` to CSS.

\* Returns a promise that resolves with a [[CompileResult]] if it succeeds and

\* rejects with an [[Exception]] if it fails.

\*

\* This only allows synchronous or asynchronous [[Importer]]s and

\* [[CustomFunction]]s.

\*

\* \*\*Heads up!\*\* When using Dart Sass, \*\*[[compile]] is almost twice as fast as

\* [[compileAsync]]\*\*, due to the overhead of making the entire evaluation

\* process asynchronous.

\*

\* @example

\*

\* ```js

\* const sass = require('sass');

\*

\* const result = await sass.compileStringAsync(`

\* h1 {

\* font-size: 40px;

\* code {

\* font-face: Roboto Mono;

\* }

\* }`);

\* console.log(result.css);

\* ```

\*

\* @category Compile

\* @compatibility dart: "1.45.0", node: false

\*/

export function compileStringAsync(

source: string,

options?: StringOptions<'async'>

): Promise<CompileResult>;